ENGINYERIA GEOMÀTICA I TOPOGRAFIA

TREBALL DE FI DE GRAU

A CERES SOLVER BASED BUNDLE ADJUSTMENT MODULE

Projectista: Andreu Alvarruiz Serrano

Directors: Joan Arnaldich Bernal, Albert Prades Valls

Convocatòria: Juny/Juliol 2014

Resum

A principis de la dècada dels 90, l'*Institut Cartogràfic i Geològic de Catalunya* va desenvolupar un programa anomenat *ACX* que permet, entre altres aplicacions, aerotriangular blocs fotogramètrics fent ús del mètode de l'ajust de feixos en bloc. Després de més de 20 anys utilitzant *ACX* a *l'ICGC* sorgeixen dues necessitats diferents. Mentre en l'àmbit de producció l'eficiència és d'especial importància, en l'àmbit de desenvolupament ho són l'exactitud, el control estadístic, la generalitat i la flexibilitat. Aquest treball neix amb l'objectiu de dissenyar el prototipus d'un nou mòdul d'ajust de feixos que satisfaci les noves necessitats del departament de desenvolupament. Es tractarà doncs, de determinar el mètode òptim per resoldre l'ajust de feixos, adaptar-lo als requeriments del flux fotogramètric de l'*ICGC* i fer que el nou mòdul, en la mesura que sigui possible, suposi una millora respecte del programari vigent.

A la primera part es donarà una visió general i teòrica del problema de l'ajust de feixos des del punt de vista actual. S'estudiaran conceptes bàsics com ara models matemàtics, mètodes d'optimització, tècniques de resolució del sistema linealitzat estratègies de resolució del sistema no lineal i tècniques de diferenciació. La primera part acaba amb un ventall de paquets que poden servir com a punt de partida per aquells lectors interessats en desenvolupar el seu propi software.

A la segona part s'analitza el programari vigent, l'*ACX*. Des d'un punt de vista teòric ens documentarem sobre els models matemàtics que té implementats. Tot seguit, a través d'un conjunt d'emulacions s'imitarà el comportament d'*ACX* per tal de detectar possibles confusions a l'hora d'interpretar els models. Els conceptes explicats a la primera part del treball serviran per comprendre més bé el funcionament d'*ACX* i alhora adquirir un criteri per triar en quines llibreries es recolzarà el nou mòdul.

Ceres Solver és el nom d'un paquet de llibreries desenvolupat a *Google* i que la mateixa empresa utilitza des de fa més de 3 anys per totes aquelles aplicacions que requereixen fer ajustos de grans sistemes no lineals pel mètode de mínims quadrats. *Ceres Solver* és propietat de *Google*, però es distribueix com a codi obert per tal de rebre retroalimentació d'usuaris d'arreu del món que informen d'errors i suggereixen possibles millores. A la documentació de *Ceres* s'adverteix que, per tal d'utilitzar-lo correctament, calen uns mínims coneixements teòrics de l'ajust per mínims quadrats. Aquest és el paquet escollit i servirà com a base per implementar un nou mòdul d'ajust de feixos en C++.

A la tercera part s'explica com utilitzar *Ceres* per resoldre problemes d'ajust d'observacions i més endavant es donen els detalls d'implementació del nou mòdul. El nou mòdul contempla l'ajust d'observacions fotogramètriques, de recolzament, mesures GPS i offset d'antena. Addicionalment contempla l'ús de paràmetres d'autocalibratge d'Ebner. Per tal de materialitzar les classes es proposen dues estructures, una plana i l'altra jeràrquica.

A la quarta part es presenta una eina anomenada *AeroSint* i desenvolupada específicament per avaluar el nou mòdul. Aquesta eina permetrà generar blocs fotogramètrics sintètics de mides i configuracions molt diverses i que contemplen tots els models considerats al nou mòdul. A més a les simulacions s'afegirà soroll gaussià de manera controlada

S'ha aconseguit desenvolupar un nou mòdul basat en *Ceres* que permet resoldre blocs d'aerotriangulació de manera eficient, utilitzant memòria dinàmica, aprofitant la flexibilitat que suposa la tècnica de diferenciació automàtica a l'hora de crear nous models, amb tractament especialitzat de matrius disperses, detecció d'errors grollers i la possibilitat d'escollir una rica parametrització. El principal aspecte que es recomana millorar és la implementació de l'estructura jeràrquica aplicant tot el potencial de la programació orientada a objectes per tal d'optimitzar la flexibilitat del nou mòdul.

Índex

Llista d'abreviacions

ACX	Ajust Combinat de Xarxes
AdIL	Adjustment Interface Language
ADS-40	Airborne Digital Sensor de Leica
ATAP	Advanced Technology and Projects
DGAP	Dirks General Analytical Positioning
DiSO	Orientació Directa de Sensors
DL	Estratègia de Dog Leg
DSM	Digital Surface Model
FORTH	Foundation for Research and Technology-Hellas
GN	Estratègia de Gauss-Newton
GENA	Generic Extensible Network Approach
GeoTeX	Geodèsia, Teledetecció i Xarxes
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
ICGC	Institut Cartogràfic i Geològic de Catalunya
IGN	Actualment conegut com a Institut National de l'Information Géographique et Forestiére
IJRR	International Journal of Robotics Research
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
InSO	Orientació Indirecta de Sensors
ISO	Orientació Integrada de Sensors
LM	Estratègia de Levenberg-Marquardt
PPS	Punt Principal de Simetria
RMS	Error mitjà quadràtic (de l'anglès Root Mean Square)
SBA	Sparse Bundle Adjustment
SFM	Structure From Motion
SLAM	Simultaneous Location And Mapping
SPOT	Satellite Pour l'Observation de la Terre
TBD	To Be Deveolped
UTM	Universal Transverse Mercator

1 Introducció i objectius

Des de l'any 1992 l'*Institut Cartogràfic i Geològic de Catalunya*, en endavant *ICGC*, disposa d'un software d'ajust d'observacions anomenat *ACX* (*Ajust Combinat de Xarxes*) que es fa servir majoritàriament per ajustar xarxes geodèsiques i blocs fotogramètrics.

ACX va ser desenvolupat a *l'ICGC* arran de la tesi doctoral del Dr. Ismael Colomina, *Structural Aspects of Hybrid Networks in Geodesy and Photogrammetry* [1], presentada el novembre de 1991. Es tracta d'un programa d'ajust molt estable i amb un tractament rigorós dels models matemàtics (funcionals i estocàstics[1]) que admet l'ús d'observacions topogràfiques, geodèsiques, fotogramètriques, *GPS*, *INS*, autocalibratge, models numèrics d'alçades, etc.

1.1 La necessitat d'un nou software. Justificació

Després de més de 20 anys utilitzant *ACX* a *l'ICGC* sorgeixen dues necessitats diferents. Una en l'àmbit de producció i una altra en desenvolupament.

Des del punt de vista de producció els resultats no necessàriament han de ser els millors possibles, simplement han de ser suficientment bons pel producte a generar. Actualment existeixen paquets de software comercials i de lliure distribució que són més eficients malgrat no siguin tan rigorosos i que podrien agilitzar la cadena de producció.

D'altra banda, al departament de desenvolupament l'eficiència no és un requeriment primordial sinó un avantatge secundari. En canvi, sí és vital que el nou software sigui exacte, genèric i, per davant de tot, flexible. *ACX* es va desenvolupar originàriament *Fortran 77*. Aquest entorn no permetia utilitzar memòria dinàmica i això implica que cal reservar memòria per avançat. Últimament a l'*ICGC* s'han fet alguns esforços per traduir part del codi a *Fortran 90* que sí permet l'ús de memòria dinàmica, però essencialment continua estant orientat a memòria estàtica.

El present treball és una proposta de millora del software existent en l'àmbit de desenvolupament. S'ha optat per iniciar un nou software que aprofiti el potencial de la programació orientada a objectes, memòria dinàmica i, en especial, l'ús de llibreries especialitzades. En els darrers anys s'ha avançat molt en el relació al problema de l'ajust de feixos. Concretament en el perfeccionament de tasques especialitzades com la resolució del sistema lineal, el tractament de matrius disperses i l'ús d'una tècnica recent i molt potent, la diferenciació automàtica [2].

Un clar exemple de que els avenços en l'ajust de feixos estan a l'ordre del dia és la recent investigació sobre quina és l'estratègia més adient per resoldre el sistema no lineal, peça clau en l'eficiència de qualsevol software que abordi el problema. L'agost de 2004 *Lourakis* i *Argyros* presenten un article [3] en què s'argumenta que l'estratègia de *Levenberg-Marquardt* amb tractament de matrius disperses s'adopta de forma quasi universal per resoldre l'ajust de feixos. En canvi, els mateixos autors publiquen a l'octubre de 2005, poc més d'un any més tard, un altre article que es qüestiona la universalitat d'aquest mètode i argumenta que l'estratègia *Dogleg* és més eficient. Més endavant, el 2010, *Konolige* proposa a [4] un refinament amb tractament "secundari" de matrius disperses que assegura superar amb escreix el plantejat per *Lourakis* i *Argyros* el 2005.

[1] Al cos de la memòria de la tesi [1] es justifica el motiu de no considerar els valors de la correlació entre observacions.

1.2 Objectius

- **Determinar el mètode òptim per resoldre el problema de l'ajust de feixos.** En aquest sentit és important aprofitar llibreries ja existents per dedicar-se a esbrinar quins són els millors mètodes en comptes d'implementar els algorismes en sí. Per exemple, de cara a resoldre el sistema no lineal la intenció és determinar quina estratègia és més convenient pel cas fotogramètric, *Levenberg-Marquardt* (*LM*), *Dogleg* (*DL*) o el clàssic de *Gauss-Newton* (*GN*).
- **Adaptar el mètode escollit als requeriments del flux fotogramètric establert a l'*ICGC*.** La necessitat de "crear" un nou software en comptes d'aprofitar programari "tal com és" ve donat per les "particularitats" del *modus operandi* de l'*ICGC*. Un clar exemple d'aquestes particularitats és l'aplicació de paràmetres d'autocalibratge distingint entre les diferents òptiques d'una càmera modular. En aquest sentit s'implementarà el codi necessari per resoldre el tipus de problemes fotogramètrics més comuns a l'*ICGC*, adaptant-se la metodologia establerta, a partir dels paquets de llibreries escollides al punt anterior.
- **Millorar el software vigent a l'ICGC.** La possible millora s'avalua en base als aspectes recollits a l'apartat *5 Avaluació del nou mòdul*:
    - Exactitud: s'espera igualar l'exactitud.
    - Estabilitat: s'espera millorar gràcies a l'adopció d'estratègies de resolució del sistema no lineal més sofisticades com ara *LM* o *DL*.
    - Control estocàstic: no s'espera superar ni igualar en aquest aspecte. Es deixarà la porta oberta a futures recerques.
    - Robustesa: s'espera certa millora gràcies a l'ús de mètodes robustos.
    - Eficiència: malgrat maximitzar l'eficiència no sigui un requeriment per desenvolupament, sí és un avantatge clar el fet de que un bloc trigui 10 minuts o bé 10 dies. En aquest sentit s'espera obtenir una millora substancial gràcies a l'aprofitament de llibreries especialitzades i a l'ús d'un programari que és propietat *Google*, però que l'empresa distribueix sota una llicència de codi obert. La coordinació i implementació de nou codi per aquest programari és responsabilitat de dos desenvolupadors de *Google*, *Sameer Agarwal* i *Keir Mierle*, però compta amb detecció d'errors i propostes de millora per part d'usuaris d'arreu del món que utilitzen el codi per les seves pròpies aplicacions.
    - Generalitat: serà l'última etapa d'implementació del software i en aquest treball no es pretén arribar a igualar *ACX*.
    - Flexibilitat: és l'objectiu principal del treball. S'esperen millores substancials gràcies a la diferenciació automàtica i la multi-parametrització. Deixar la porta oberta a l'ampliació del codi per considerar nous models i nous problemes. En aquest sentit cal que les llibreries escollides siguin prou genèriques i el codi desenvolupat prou flexible.

2 Part I. Recerca

2.1 L'ajust de feixos. Perspectiva actual

L'ajust de feixos[2] és un dels mètodes d'optimització que permet determinar les coordenades tridimensionals d'un conjunt de punts a partir de mesures sobre imatges bidimensionals alhora que s'estimen els paràmetres d'orientació i calibratge més probables dels sensors. Actualment existeixen dues disciplines que fan ús d'aquesta tècnica: la comunitat fotogramètrica i el món de la visió per computador [5].

Des del punt de vista fotogramètric, el mètode d'ajust de feixos és un dels motors de càlcul més utilitzats per resoldre un problema més genèric: l'orientació i calibratge de sensors. A continuació es presenten algunes de les tècniques fotogramètriques actuals:

- Orientació Indirecta de Sensors (*InSO*): ajust de feixos clàssic sense recolzament aeri. Únicament s'ajusten observacions fotogramètriques i recolzament terrestre. El resultat obtingut té poca fiabilitat.
- Orientació Integrada de Sensors (*ISO*): es combinen observacions de diferents fonts, en general s'ajusten observacions fotogramètriques, recolzament terrestre, recolzament aeri (GNSS i INS), etc. És el mètode que dóna major fiabilitat i també és l'escollit per la majoria de línies de producció de fotogrametria actuals.
- *Minimal AT*: variant del mètode *ISO* proposada a [6] que consisteix en prescindir de la majoria de punts de lligam, conservant només els de passades transversals i pels punts de suport.
- *Fast AT*: variant proposada a [7] on només s'incorporen observacions de recolzament aeri, recolzament terrestre i les observacions fotogramètriques pels punts de recolzament. Tant l'exactitud i fiabilitat com el rendiment es troben a cavall dels mètodes *ISO* i *DiSO*.
- Orientació Directa de Sensors (*DiSO*): l'orientació del sensor es determina únicament a partir del recolzament aeri. El preu a pagar és la baixa fiabilitat, però en la majoria de casos podria ser suficient pels requeriments del producte. Aquest és un exemple de mètode d'orientació de sensors que no utilitza el mètode d'ajust de feixos pròpiament.

Entitats com ara *l'Institut de Geomàtica* i empreses com *Geonumerics* han desenvolupat recentment nous softwares d'ajust de xarxes i justifiquen a [8] i a [9] que, malgrat la incorporació de tècniques com GNSS i INS, l'ajust de xarxes no està en absolut obsolet i encara menys estancat. Al contrari, argumenten que la integració de gran quantitat d'informació en els mòduls d'ajust ha posat a prova els mètodes i tècniques d'ajust actuals.

A continuació es presenten dues disciplines de la visió per computador, *SFM* i *SLAM*.

Structure From Motion (*SFM)* és el nom d'una de les disciplines de visió per computador que permet reconstruir una escena tridimensional a partir d'imatges bidimensionals. Tant *SFM* com el mètode d'aerotriangulació en fotogrametria es basen en el mètode d'ajust de feixos per obtenir resultats. L'objectiu de *SFM* és obtenir els punts objecte, en canvi, l'objectiu de l'aerotriangulació és obtenir l'orientació dels sensors. Entre altres aplicacions, *SFM* obté núvols de punts densificats que permeten generar escenes tridimensionals fotorealistes. Entre vàries aplicacions conegudes de

[2] En la literatura anglosaxona, *bundle adjustment* i en la germànica, *Bündelausgleichung*

SFM destaquen *Street View* [10] de *Google*, el projecte *Building Rome in a day* [11], el software *Photosynth* de *Microsoft*, etc.

Simultaneous Location And Mapping (*SLAM*) és un cas particular de *SFM* enfocat a la robòtica que també utilitza l'ajust de feixos com peça clau. Consisteix en fer que el mateix sensor que captura la geoinformació per fer cartografia, també sigui capaç de conèixer la seva pròpia posició i actitud. La dificultat rau en el fet de demanar al propi sensor que conegui la posició sobre un mapa que ell mateix ha de crear. *OpenSLAM* és una iniciativa iniciada el gener de 2007 que promou la cooperació entre desenvolupadors i amb la intenció d'unir esforços. Un exemple molt recent de l'ús de l'ajust de feixos en *SLAM* és un article presentat l'agost de 2013 a la *IJRR* [12] en què es presenta un nou software anomenat *ParallaxBA*.

Una aplicació de *SLAM* i que utilitza *Ceres Solver* com a motor d'ajust de feixos és el projecte *Tango* [13]. Es tracta d'una iniciativa de *l'Advanced Technology And Projects* (*ATAP*) de *Google* que va sortir a la llum al febrer de 2014 i que encara està en fase experimental. S'han distribuït 200 prototipus d'un dispositiu mòbil que té dues càmeres i un distanciòmetre integrats amb l'objectiu d'obtenir un model tridimensional de l'entorn a mesura que l'usuari es mou. La intenció de *Tango* és que qualsevol usuari d'un dispositiu mòbil contribueixi a generar una cartografia global.

A *Triggs et al.* [5] es defensa que el mètode de l'ajust de feixos és tant o més eficient que la majoria d'algorismes de visió per computador, alhora que és flexible, en el sentit que pot manegar gran varietat de models, i acurat, ja que permet un control estocàstic rigorós. Només cal analitzar acuradament l'estructura del problema, escollir els mètodes de resolució més adients i fer un tractament adequat de matrius disperses. En el mateix article s'anima a la comunitat de la visió per computador a "unir-se al tren" de la comunitat fotogramètrica i avançar junts per millorar el mètode d'ajust de feixos.

Segons *Agarwal, Mierle et al.* [2], en qualsevol software d'ajust d'observacions és convenient que el mètode de resolució del sistema sigui independent de com es defineixin els models.

- Models matemàtics: funcionals i estocàstics
- Mètode escollit per l'optimització
- Estratègia de resolució del sistema no lineal
- Mètode de resolució del sistema linealitzat

Aquesta arquitectura modular és una peça clau que permetrà que un software d'ajust sigui flexible i extensible. A [9] es proposa un exemple concret d'arquitectura modular orientada a flexibilitzar la incorporació de nous models matemàtics, p.e. la incorporació d'un nou instrument de mesura no hauria d'afectar als mòduls que es dediquen a resoldre l'ajust.

2.2 Models matemàtics

Genèricament, els models matemàtics defineixen el comportament d'un sistema utilitzant llenguatge matemàtic [8]. En el cas particular de l'ajust d'observacions, es poden considerar dos models matemàtics, el funcional i l'estocàstic:

2.2.1 Model funcional

El model funcional defineix la relació entre un conjunt *U* de *m* observacions

$$U = \{u_1, u_2 ... u_m\}$$

i un conjunt *X* de *n* paràmetres

$$X = \{x_1, x_2 ... x_n\}.$$

Aquesta relació es materialitza per mitjà de les equacions del sistema

$$F = \{f_1, f_2 ... f_k\} \ \mid \ f_i(U, X) = v_i, \ \ \forall i = 1..k$$

on f es coneix com la funció de cost[3] que permet conèixer la discrepància v entre les observacions U i la predicció que fa el model en base a certs paràmetres X. En problemes de mínims quadrats es tractarà de trobar els paràmetres que minimitzin la suma quadràtica de les funcions de cost. La definició de f vindrà donada segons el tipus de model funcional escollit:

- Models implícits: és un model genèric en què tant observacions com paràmetres intervenen a la funció de cost. El valor del cost és directament la discrepància v.

  $$f(U, X) = v_i$$

  En problemes de mínims quadrats caldrà adoptar el mètode general.

- Models explícits: és un cas particular en què es poden aïllar les observacions dels paràmetres i reescriure les funcions de cost com:

  $$F = \{f_1, f_2 ... f_m\} \ \mid \ f_i(X) = u_i + v_i, \ \ \forall i = 1..m$$

  on m és el nombre d'observacions que coincideix amb el nombre de funcions de cost (i per tant, amb el nombre d'equacions del sistema).

  En problemes de mínims quadrats serà possible escollir el mètode d'observacions indirectes o equacions d'observació.

2.2.2 Model estocàstic

El model estocàstic descriu la distribució de probabilitat d'una variable aleatòria que intervé en l'ajust d'observacions. Concretament, les variables aleatòries consistiran en observables afectats per soroll gaussià. El model estocàstic permet ponderar les observacions en funció de les desviacions associades, fer un anàlisi de la propagació d'errors i, finalment, conèixer la qualitat de l'ajust resultant.

Per cada una de les m observacions

$$U = \{u_1, u_2 ... u_m\}$$

es coneixen les desviacions corresponents

$$\sigma_{xx} = \{\sigma_{11}, \sigma_{22} ... \sigma_{mm}\}$$

A més, per cada parell d'observacions m,p les correlacions seran:

$$\sigma_{xy} = \{\sigma_{12}, \sigma_{13}, ..., \sigma_{21}, \sigma_{23}, ..., \sigma_{mp}\}$$

Genèricament el pes p_i d'una observació u_i vindrà donat per la inversa de la seva desviació σ_i:

$$P = \{p_1, p_2 ... p_m\}$$

Els anomenats mètodes robustos modifiquen el pes de certes observacions mitjançant una funció $\rho(\cdot)$ que "envolta" la funció de cost $f(U,X)$:

---

[3] Molt sovint en la literatura anglosaxona també rep el nom de *objective function* o *loss function*.

$$\rho(f(U,X))$$

La funció ρ (·) farà disminuir el cost d'una observació amb residus per sobre de cert llindar, reduint així el pes a l'ajust d'aquelles observacions que continguin errors grollers.

2.3 Mètode d'optimització

En general, l'optimització de la funció de cost consisteix en determinar el vector de paràmetres *X* que satisfaci la condició de màxima versemblança:

$$\text{Donat } F:\mathbb{R}^2 \rightarrow \mathbb{R}, \text{ cal determinar } X = \arg\max_x \{f(U,X)\}$$

Què és equivalent a minimitzar la funció de cost:

$$X = -\log(f(U,X))$$

El mètode de mínims quadrats és un cas particular en què es defineix:

$$F(x) = \frac{1}{2}\sum_{i=1}^{m}(f_i(U,X))^2$$

Cal doncs determinar un vector $X = \{x_1, x_2 ... x_n\}$ que minimitzi $F(x)$. En general, trobar una solució global és molt difícil i a la pràctica es determina un mínim local assumint que es parteix d'una aproximació prou propera al mínim global. Llavors, el mínim local es pot calcular igualant el gradient a zero (evidentment assumint que és un mínim i no un màxim):

$$\nabla = F'(x) = 0$$

A continuació es presenten dos dels mètodes d'ajust més habituals que permeten resoldre el problema d'ajust de feixos. Escollir un o altre mètode dependrà de la possibilitat de definir els models funcionals com explícits o implícits.

2.3.1 Mètode general dels mínims quadrats

En el mètode general, les funcions de cost dependran tant de paràmetres com d'observacions. Per tant, els models funcionals que caldrà definir seran models implícits. Un model implícit està composat per un conjunt d'equacions f_i en què s'expressa la relació entre un subconjunt de *m* observacions i *n* paràmetres de la forma:

$$f_i(o_1, o_2 ..., o_m, x_1, x_2 ..., x_n) = v_i$$

La funció f_i retorna un escalar v_i que serà directament el cost de considerar els paràmetres $x_1, x_2 ..., x_n$ i les observacions $o_1, o_2 ..., o_m$ en el model escollit.

El sistema linealitzat tindrà la forma següent:

$$AX + BV + W = 0$$

on *A* és la jacobiana de *f* respecte dels paràmetres $X = \{x_1, x_2 ..., x_n\}$ i *B* és la jacobiana de *f* respecte de les observacions $U = \{u_1, u_2 ..., u_m\}$ amb residus $V = \{v_1, v_2 ..., v_m\}$. *W* és el vector de termes independents (en anglès *misclosures*).

2.3.2 Mètode d'equacions d'observació

És un cas particular del mètode general en què és possible aïllar cada observació o_i del model i expressar-la com a funció d'un subconjunt de *s* paràmetres $x_1, x_2 ..., x_s$.

Llavors tenim que les funcions de cost les podem expressar en la forma:

$$o_i + v_i = f_i(x_1, x_2..., x_s)$$

Aquest cop la funció f_i retorna un escalar que serà una observació o_i més el cost v_i.

El sistema linealitzat continuarà tenint la forma anterior, però en aquest cas *B* serà la identitat i, per tant, es pot reescriure com:

$$Ax + v + w = 0$$

Aquest mètode no considera la possible correlació entre observacions. Una explicació detallada del mètode d'equacions d'observació es pot trobar a [14].

2.3.3 Quin mètode és més adient per resoldre l'ajust de feixos?

Per una banda, la resolució de l'ajust considerant el mètode d'equacions d'observació és més simple ja que només cal considerar una jacobiana, la dels paràmetres. Però per altra banda, no serà possible considerar la correlació entre observacions mitjançant una matriu de variància-covariància a priori. En aquestes condicions no es podrà aïllar cada observació i expressar-la només en funció de paràmetres.

En el problema corresponent a l'ajust de feixos el model principal ve donat per la condició de col·linealitat. És a dir, el model relaciona les fotocoordenades $\{x, y\}_i$ d'un punt *p* amb les seves coordenades objecte $\{X, Y, Z\}_p$ i les orientacions externes $f_i(\{X, Y, Z, w, p, k\}_f$ de la imatge on s'observa.

$$\{x, y\}_i = f_i(\{X, Y, Z, w, p, k\}_f, \{X, Y, Z\}_p)$$

Si les úniques observacions a considerar en l'ajust de feixos fossin les observacions fotogramètriques, llavors directament es podrien definir els models en forma explícita i, per tant, fer servir el mètode d'equacions d'observació. Fins aquí la tria està clara. Els dubtes sorgeixen a l'hora de considerar altres paràmetres de les equacions de col·linealitat com a observació.

Què passa quan es decideix considerar les coordenades dels punts de recolzament com a observacions? Si l'ajust només considera el model de col·linealitat aleshores deixa de ser possible aïllar directament les fotocoordenades de tots els punts observats.

Per exemple, suposem que a la imatge *i* mesurem dos punts. El punt 1 és de traspàs i el 2 és de recolzament. Llavors a les funcions de cost

$$f(\{X, Y, Z, w, p, k\}_i, \{X, Y, Z\}_1)$$

$$f(\{X, Y, Z, w, p, k\}_i, \{X, Y, Z\}_2)$$

$\{X, Y, Z\}_1$ i $\{X, Y, Z\}_2$ seran vectors de paràmetres a determinar, però $\{X, Y, Z\}_2$ a més actuarà com a variables observades amb certa desviació tipus associada. La jacobiana de *f* respecte de les observacions deixaria de ser la identitat i, per tant, el mètode de resolució a triar seria el mètode general dels mínims quadrats.

A l'apartat 3 de la memòria, *Anàlisi del software vigent a l'ICGC,* s'exposa l'alternativa utilitzada a *ACX* [1] segons la qual, en el cas fotogramètric, és possible considerar que la jacobiana respecte de les observacions és la identitat partint del mètode general de mínims quadrats.

2.4 Estratègia de resolució del sistema no lineal

Segons *Madsen* [15] qualsevol estratègia de resolució d'un sistema no lineal es basa en un algorisme iteratiu que fa convergir els paràmetres cap una la solució que minimitzi la funció de cost.

Els anomenats mètodes descendents imposen la restricció de que per a cada iteració el cost disminuirà respecte de l'anterior. Els *passos* resultants seguiran el progrés corresponent a una funció monòtona decreixent. D'altra banda, existeixen els mètodes no descendents, que a *Agarwal* i *Mierle* [2] anomenen *non-monotonic step methods* (mètodes en que la funció de variació del valor del pas no es restringeix a ser monòtona decreixent). Eliminant aquesta restricció es pretén arribar més ràpid a la solució final permetent que el cost pugui augmentar en certes iteracions. A [2] se cita un exemple molt il·lustratiu: cal imaginar-se l'espai de paràmetres com un paisatge de muntanya on es permet que l'algorisme "salti per sobre de roques" i que no s'hagi d'endinsar en valls estretes que podrien fer perdre el rumb que portava.

Tornant als mètodes descendents, aquests parteixen d'una aproximació inicial i, per cada iteració, es calculen dos paràmetres:

- *Direcció de cerca descendent:* cal buscar un vector de l'espai de paràmetres que faci disminuir la funció de cost el proper cop que s'avaluï.
- *Pas*: provinent del terme anglès *step* és el mòdul del vector *direcció de cerca descendent*, és a dir, la magnitud de la correcció que s'aplicarà als paràmetres.

A continuació es presenta una proposta d'algorisme en pseudocodi per implementar una estratègia genèrica de resolució de sistema no lineal. Només es tracta d'una simplificació per visualitzar el procediment.

```
x := x0          # Inicialització del vector de paràmetres
tol := 1.0E-6    # Màxima correcció als paràmetres
max_iter := 30 # Màxim nombre d'iteracions
iter := 1        # Iteració actual
pas := 9999      # Magnitud de correccions actual

while (pas > tol) and (iter < max_iter)
   dir := my_strategy.search_dir(x) #Actualitza direcció de cerca
   pas := my_strategy.pas(x)    # Actualitza el pas
   x := x + pas*dir          # Actualitza vector de paràmetres
   iter := iter + 1          # Següent iteració

if (pas > tol)
   mostra( " Solution converged: parameter tolerance reached " )
else
   mostra( " No convergence " )
```

En el procés iteratiu d'un mètode descendent es distingeixen dues fases:

- Fase global: es fa una cerca "a grans trets" de la direcció descendent. El mètode de cerca més eficient en aquesta fase acostuma a ser el mètode del gradient[4]. Aquest mètode pren com a direcció de cerca el vector gradient.
- Fase final: es busca "amb precisió" el mínim de la funció de cost quan el procés s'apropa a la solució. En aquesta fase en canvi, seria més eficient el mètode de *Newton*.

[4] Conegut en anglès com a *Gradient Descent* o també *Steepest Descent*

Un exemple d'estratègia eficient podria consistir en utilitzar el mètode del gradient en la fase global i canviar al mètode de *Newton* quan el procés està arribant a la solució final. El primer mètode garanteix la convergència i el segon és molt més eficient de cara a afinar la solució.

Els mètodes de *Newton*, gradient, cerca unidimensional[5], regió de confiança[6], amortiment[7], etc. i variants o combinacions seran els que s'utilitzaran a les diferents estratègies de resolució del sistema no lineal.

En general, la diferència entre estratègies de mètodes descendents és l'elecció dels paràmetres *direcció de cerca descendent* i la magnitud del *pas*. A continuació es presenten les estratègies de resolució més comunes. No s'inclou la demostració de com s'arriba al sistema lineal a partir de la funció a minimitzar, només es mostra l'essència de cada estratègia per diferenciar-la de les altres. Per una demostració en detall es pot consultar [15] o [16].

2.4.1 Estratègia de *Gauss-Newton*

L'estratègia de *Gauss-Newton* (*GN*) és molt eficient quan els valors dels paràmetres estan relativament a prop de la solució. Es basa en l'ús de la primera derivada de la funció de cost, és a dir, es calcula un desenvolupament en sèrie de *Taylor* de primer ordre per obtenir una aproximació lineal del model centrat en el valor dels paràmetres per cada iteració.

Assumint un valor prou petit de *dx* es pot considerar:

$$f(x+dx) = f(x) + \frac{\partial f(x)}{\partial x} dx$$

El valor del *pas* ve donat per la solució *dx* al sistema lineal que minimitza el cost *f(x)*:

$$(A^t PA)\, dx = A^t PL$$

on $A = \frac{\partial f(x)}{\partial x}$ és la jacobiana de *f* respecte del vector de paràmetres *x*

$P = C^{-1}$ és la inversa de la matriu de covariància

i $L = f(x)$ és la funció de cost o residu

GN és l'estratègia clàssica de resolució de sistemes no lineals. Un dels inconvenients és que pot divergir si l'aproximació inicial no és prou propera al mínim global de la funció de cost.

2.4.2 Estratègia de *Levenberg-Marquardt*

Segons s'argumenta a [2], l'estratègia de *Levenberg-Marquardt* (*LM*) és més robusta en el sentit que trobarà una solució malgrat l'aproximació inicial es trobi molt lluny de la solució. Com a contrapartida, *LM* serà una mica més lent que *GN* si l'aproximació inicial és prou bona. Es tracta d'una lleugera modificació del mètode de *GN* que augmenta la possibilitat de convergència cap al mínim local i, a més, fa que la solució mai arribi a divergir. Això sí, al igual que en el mètode *GN*, el mínim cap al que convergirà no necessàriament serà el global. *LM* busca una adaptació gradual entre el mètode del gradient i el de *GN* al llarg del procés iteratiu.

---

[5] Conegut en anglès com a *Line-Search method*
[6] Conegut en anglès com a *Trust region method*
[7] Conegut en anglès com a *damped methods* en base al factor d'amortiment que utilitzen.

Al mètode original de *Levenberg* la modificació ve donada pel fet d'introduir un coeficient d'amortiment μ a la matriu normal *N* del sistema linealitzat.

$$(A^t PA + \mu\, I)\, dx = N\, dx = A^t PL$$

Marquardt proposa una petita variant del mètode original de *Levenberg* que consisteix en substituir la matriu identitat *I* per la diagonal de la matriu normal:

$$(A^t PA + \mu\, diag\,(A^t PA))\, dx = N\, dx = A^t PL$$

D'aquesta manera la matriu normal s'escala d'acord amb la magnitud dels paràmetres. Segons [16] aquesta millora comporta una convergència més ràpida que la versió original en cas de tenir grups de paràmetres de magnituds molt diferents. L'ajust de feixos és un clar exemple ja que existiran blocs d'observacions d'escales molt diferents.

Sigui quina sigui la variant, *Levenberg* o *Levenberg-Marquardt*, el coeficient d'amortiment μ actua com a factor d'escala de la matriu normal $A^t A$ i anirà variant el seu valor al llarg del procés iteratiu per tal d'adaptar l'estratègia a la proximitat dels paràmetres a la solució [15].

- Sempre que $\mu > 0$ la matriu normal serà definida positiva i això garanteix que *dx* segueixi una direcció descendent
- Si μ és gran, el *pas* resultant serà petit i la direcció de cerca tendirà cap a la del gradient.
- Si μ és petit, el *pas* s'aproparà al del mètode de *GN*.

L'actualització de μ es fa automàticament en base a un coeficient β que indica la millora de la predicció. És a dir, la disminució del vector de residus respecte dels residus inicials:

$$\beta = \frac{L(x) - L(x + dx)}{L(x_0) - L(dx)}$$

Un cop determinat el coeficient de millora β, s'obté el coeficient d'amortiment μ segons els següents criteris:

- Si β és petit vol dir que l'aproximació és dolenta i llavors cal augmentar μ per disminuir la mida del *pas* i tornar a la direcció de cerca del gradient.
- Si β és prou gran vol dir que l'aproximació és bona i llavors cal disminuir μ per apropar-se al mètode de *GN*.

Per més detalls, a [15] es pot consultar l'algorisme de *LM* implementat en pseudocodi i a [17] el mateix, però en *C/C++*. A [3] es pot consultar la implementació d'una aplicació del mètode de *LM* pel problema concret de l'ajust de feixos.

2.4.3 Estratègia de *Powell's Dog leg*

El mètode *Dog leg* (*DL*), proposat per Powell, també és una combinació entre els mètodes de *GN* i *SD* que consisteix en trobar una direcció de cerca a partir dels vectors donats pels *passos* dx_{GN} i dx_{CY} respectivament i, a més, restringit per una *regió de confiança* de radi R_{TR}. Es pot trobar una descripció detallada d'aquest mètode a [15].

Com s'ha explicat a l'estratègia de *Gauss-Newton*, el *pas* dx_{GN} es pot calcular resolent el sistema lineal:

$$(A^t PA)\, dx_{GN} = A^t PL$$

El vector dx_{CY} s'obté en minimitzar la funció de cost $f(x)$ restringint-se a la direcció del gradient $\nabla = A^t PL$:

$$dx_{SD} = \frac{||\nabla||^2}{||A\nabla||^2}\nabla$$

L'algorisme per obtenir el *pas* dx_{SD} que es proposa a [15] és el següent:

```
if (abs(dxGN) <= R)
      dxDL := dxGN
else if (abs(dxCY) >= R)
      dxDL := (R/abs(dxCY))dxCY
else
      dxDL := dxCY + s(dxGN - dxCY)
```

on el coeficient *s* s'escull per tal que: $\| dx_{DL} \| = R_{TR}$

A la figura 2.1 s'il·lustra gràficament el "camí" que segueix l'estratègia en el 3er cas de l'algorisme anterior on el *pas* dx_{DL} pren la mida i direcció del vector en negreta, és a dir:

$$\| dx_{DL} \| = dx_{DL} + s(dx_{GN} - dx_{CY}) = R_{TR}$$

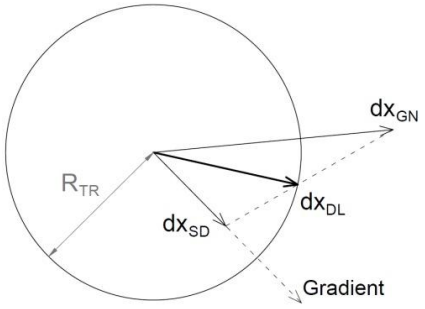


Figura 2.1

Segons [18] a l'hora de resoldre el problema de l'ajust de feixos, el mètode *Dog leg* redueix considerablement el cost de computació en comparació amb l'ús del mètode *LM*.

A [2] s'explica l'avantatge principal del mètode *DL* respecte de *LM* que fa que sigui més eficient: en cas que el *pas* escollit no comporti una disminució prou gran de la funció de cost, el mètode *LM* torna a començar de zero, però escollint un valor de coeficient d'amortiment μ més gran. En canvi, el mètode *Dog leg* es limita a calcular un *pas* interpolat entre el vector de *GN* i el de *Cauchy*.

2.5 Mètode de resolució del sistema linealitzat

Es tracta de l'algorisme emprat per resoldre el sistema lineal corresponent a cada iteració de l'estratègia del sistema no lineal. La solució donarà com a resultat el *pas* necessari per progressar en el sistema iteratiu. En els mètodes més habituals la determinació del *pas* implicarà un càlcul directe, en altres més complexos com el mètode iteratiu de *Schur*, caldrà fer iteracions per obtenir el *pas*.

Tal i com es diu a [5], per resoldre el sistema lineal gairebé mai s'arriba a invertir la matriu normal. En moltes fórmules apareixen matrius inverses de cara a simplificar la notació, però a la pràctica s'utilitzen mètodes de factorització. A continuació es presenten alguns de les alternatives

més habituals. Els costos de computació es consideren per una matriu normal de mida $n_x n$, on n és la dimensió del vector de solució.

- **Eliminació pel mètode de Gauss**: és el mètode acadèmic per resoldre un sistema lineal d'equacions. En paraules de [17], la seva utilitat principal és la pedagògica. És simple i té un bon comportament per sistemes en què la matriu del sistema és d'ordre petit. El cost de computació és $\frac{n^3}{3}$ per obtenir la solució al sistema i $\frac{4n^3}{3}$ en cas de voler obtenir a més la inversa.
- **Eliminació pel mètode de Gauss-Jordan**: els avantatges d'aquest mètode són l'estabilitat i el fet d'obtenir simultàniament la solució al sistema i la matriu inversa amb un cost de computació semblant al de mètodes més sofisticats com la descomposició *LU*. El cost de computació és n^3. Evidentment, si l'únic que interessa és obtenir la solució al sistema, llavors aquest mètode és 3 vegades més lent que el d'eliminació de Gauss [17].
- **Descomposició *LU***: consisteix en factoritzar la matriu del sistema N en dues matrius L (triangular inferior) i U (triangular superior) tals que $N = LU$. El procés per obtenir la solució al sistema $Nx = b$ es desglossa en tres passos:
  - Factoritzar N per obtenir L i U: el cost és $\frac{2n^3}{3}$
  - Calcular y al sistema $Ly = b$: el cost és n^2
  - Calcular x al sistema $Ux = y$: el cost és n^2

  Per tant el cost total és d'ordre $\frac{2n^3}{3}$. De primeres sembla que la suma dels costos és superior a la dels mètodes anteriors. L'avantatge d'aquest mètode consisteix en el fet de que un cop feta la descomposició, es pot resoldre de nou el sistema $Nx = b$ amb nous vectors de termes independents b sense haver de començar de zero.
- **Descomposició de Cholesky**: Una variant del mètode de descomposició *LU* en cas de que N sigui simètrica i definida positiva. Tenint en compte aquesta simetria, el cost de computació per factoritzar $N = LL$ es pot reduir a la meitat respecte del mètode genèric *LU*, és a dir $\frac{n^3}{3}$. Aquest és un dels mètodes més utilitzats per resoldre el sistema lineal en el problema d'ajust de feixos ja que la matriu normal per definició serà simètrica i definida positiva.
- **Descomposició QR**: Una altra variant utilitzada en l'ajust de feixos és una descomposició de $N = QR$ on R és triangular superior i Q és ortogonal, és a dir: $Q^t = Q^{-1}$. En aquest cas, resoldre el sistema original $Nx = b$ és equivalent a resoldre $Rx = Q^t b$. El cost és el doble que la descomposició *LU*, és a dir: $\frac{4n^3}{3}$. L'avantatge d'aquest mètode es fa evident quan cal resoldre un seguit de sistemes lineals en què la matriu normal canvia "poc". Segons [17], en comptes d'haver de resoldre el sistema des de zero cada cop (implicant un cost d'ordre n^3), permetria limitar-se a actualitzar les matrius Q i R amb un cost resultant d'ordre n^2.
- **Complement de Schur:** es tracta d'un dels mètodes que segons [2] més bé s'adapten a l'estructura particular de les matrius de l'ajust de feixos. Aprofita el fet que per cada funció de cost només intervenen les coordenades d'un sol punt objecte. Això fa que la matriu normal $N = A^t PA$ tingui una estructura de blocs com la següent:

$$N = \begin{bmatrix} B & E \\ E^t & C \end{bmatrix}$$

Llavors el sistema d'equacions normals es pot escriure com:

$$\begin{bmatrix} B & E \\ E^t & C \end{bmatrix} \begin{bmatrix} dx \\ dy \end{bmatrix} = \begin{bmatrix} Tx \\ Ty \end{bmatrix}$$

Ara només cal aplicar el mètode d'eliminació de Gauss al sistema anterior i així obtenir *dy*. Coneixent *dy* es pot expressar *dx* d'aquesta manera:

$$(B - EC^{-1}E^t)dx = Tx - EC^{-1}Ty$$

on $S = B - EC^{-1}E^t$ es coneix com el complement de Schur de *C* en *N*. Tenint en compte que la submatriu *C* és diagonal per blocs, calcular C^{-1} és poc costós. A més, el nombre de punts serà molt menor que el nombre d'orientacions així que, en general, obtenir *S* serà molt més ràpid que resoldre el sistema d'equacions normals original.

L'explicació anterior està basada en [2]. Per una descripció més rigorosa i detallada del complement de Schur consulteu [19].

- **Inversió parcial:** com s'ha dit anteriorment, invertir la matriu del sistema normal *N* no és la manera més eficient d'obtenir la solució als sistemes lineals. Malgrat això, un cop finalitzat el procés iteratiu, s'haurà d'obtenir la informació estocàstica dels paràmetres resultants. Per fer-ho, una alternativa és fer una inversió parcial que permeti obtenir els elements de la inversa necessaris sense haver de invertir completament la matriu *N* [1].

2.6 Tècniques de diferenciació

Bàsicament existeixen quatre tècniques que permeten obtenir la derivada d'una funció respecte de les seves variables:

- **Diferenciació manual:** es tracta d'obtenir l'expressió de la derivada a partir d'una font externa al programa i després el mateix programador la introdueixi al codi. Pot ser derivant a mà, amb calculadores programables o softwares de càlcul simbòlic, de la literatura fotogramètrica, etc. El mòdul resultant serà lleugerament més eficient (no ha de calcular la derivada) a expenses de ser poc flexible (per incloure nous models caldrà tornar a derivar les noves funcions).
- **Diferenciació numèrica:** es basa en l'ús del mètode de diferències finites. Donat un paràmetre o vector de paràmetres *x* i la funció de cost $f(x)$ es calcula el valor de $f(x)$ i el valor de $f'(x)$ segons la definició de la derivada en un punt:

  $$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

  on *h* és un valor real positiu i prou "petit".

  Segons es diu a l'apartat *5.7 Numerical derivatives* de [17], l'elecció d'un valor per *h* "petit" no és trivial ja que cal tenir en compte els errors de truncament del desenvolupament en sèrie de Taylor i els errors d'arrodoniment com a conseqüència de la representació aproximada de nombres reals com a binaris. A [17] es recomana doncs, escollir *h* tal que la diferència entre $x + h$ i *x* correspongui a un nombre representable de forma exacta en binari.

- **Diferenciació simbòlica:** és el mètode que utilitzen els programes de càlcul simbòlic com *Maple* per representar la derivada d'una expressió $f(x)$ en forma d'una altra expressió $f'(x)$ tal que $f'(x) = \frac{df(x)}{dx}$. Es tracta d'aplicar les regles bàsiques de diferenciació de caire acadèmic. És a dir, desglossar l'expressió inicial en sub-expressions, aplicar les fórmules bàsiques a cada sub-expressió i finalment aplicar la regla de la cadena per combinar composició de funcions. Per exemple, partint de l'expressió $ax^2 + bx + c$ l'algorisme hauria de retornar $2ax + b$.

  D'acord amb el mètode presentat a [20], el problema es divideix en tres parts que s'il·lustraran sobre l'exemple $ab + c^d$:

  - Interpretació de l'expressió: a [20] es presenta un pseudocodi complert i molt simple de com implementar la interpretació. En l'exemple, s'obtindrien les tres expressions següents: $e_1 = ab$, $e_2 = c^3$ i $e_3 = e_1 + e_2$.

  - Aplicar les fórmules bàsiques de diferenciació: aplicar fórmules de derivades i regla de la cadena. En l'exemple, anterior:

$$e'_1 = b\frac{da}{dx} + a\frac{db}{dx} \qquad e'_2 = 3c^2\frac{dc}{dx} \qquad e'_3 = \frac{de'_1}{dx} + \frac{de'_2}{dx}$$

  - Optimitzar l'expressió resultant: neteja de l'expressió per eliminar operadors innecessaris (parèntesis redundants, multiplicacions per 1,etc).

- **Diferenciació automàtica:** es tracta d'una alternativa recent i poc coneguda per obtenir la derivada d'una expressió que es basa en l'ús de tipus genèrics de dades, els *templates*. De fet, segons [21] la diferenciació automàtica es pot implementar en qualsevol llenguatge de programació que permeti definir un nou tipus de dades numèriques i les seves operacions bàsiques (i.e. sumes, restes, multiplicacions i divisions).

  El nou tipus es correspon a un concepte matemàtic similar al dels nombres imaginaris, els nombres duals. Segons [22] un nombre dual *z* té la forma:

$$z = a + b\varepsilon$$

  on $a, b \in \mathbb{R}$ i ε és un nou element tal que $\varepsilon^2 = 0$ i $\varepsilon \neq 0$. Llavors, el desenvolupament en sèrie de Taylor de la funció $f(x)$ es pot simplificar i representar-se de forma exacta de manera que:

$$f(x + \varepsilon) = \sum_{n=0}^{\infty} \frac{f^{(n)}(x)}{n!}\varepsilon^n = f(x) + \varepsilon f'(x)$$

  on $a = f(x)$ i $b = f'(x)$. D'aquí cal quedar-se amb el coeficient *b*.

  Els principals avantatges de la diferenciació automàtica segons [21] són:

  - Menor cost de computació que la diferenciació numèrica
  - Més exactitud que amb la diferenciació numèrica. Concretament es calcula la derivada "exacta" (igual que en la diferenciació simbòlica).
  - Més eficient que la diferenciació simbòlica
  - Manteniment i extensió del codi molt poc costós

2.7 Llibreries, paquets i softwares actuals

En els darrers anys estan sorgint nous softwares, tant comercials com de codi obert, que permeten resoldre el problema de l'orientació de sensors fent servir com a motor l'ajust de feixos. A continuació es presenta una selecció de llibreries, paquets i softwares dels quals caldrà escollir-ne un, aquell que més s'adapti als requeriments de desenvolupament de l'*ICGC*. Els criteris de cerca són: codi obert, genèric, flexible, robust, rigorós, suficientment eficient i sobretot extensible ja que la intenció és que s'adapti a la incorporació de nous sensors i noves tècniques.

- ***SBA*:** és un paquet implementat en C/C++ que permet resoldre problemes genèrics d'ajust de feixos amb tractament de matrius disperses i utilitzant l'estratègia de *Levenberg-Marquardt* [3][23][24]. Els autors són dos investigadors de l'*Institute of Computer Science* del *Foundation for Research and Technology-Hellas* (*FORTH*), Grècia. Permet escollir entre quatre maneres diferents de calcular la jacobiana: manual (picant les derivades directament al codi), amb una eina auxiliar de diferenciació simbòlica, fent una aproximació per diferències finites i, per últim, per tècniques de diferenciació automàtica. El gran inconvenient per aprofitar aquest paquet com a base pel nou mòdul és que, pel que es pot deduir de l'explicació donada a [3][23], només permet incloure càmeres i observacions fotogramètriques. És genèric en el sentit de permetre a l'usuari "inventar-se" la seva pròpia càmera i fixar les coordenades dels punts de recolzament, però aparentment no dóna la possibilitat de fer que els punts de recolzament també siguin considerats com a observacions.

- ***DGAP*:** *Dirk Stallmann* de l'*Institut für Photogrammetrie Stuttgart* ha desenvolupat un software que implementa el mètode d'ajust de feixos pel cas fotogramètric. Es tracta d'un software força genèric: destaca la possibilitat d'escollir entre diferents grups de paràmetres d'autocalibratge (*Brown*, *Ebner* i *Grün*), integració de sensors com *GPS* i *IMU*, possibilitat de fer orientació directa, incorporació de models per sensors lineals com ADS-40 i tractament detallat del model estocàstic [25]. Per una explicació detallada de *DGAP* es recomana consultar el manual d'usuari que es troba a [26].

- ***Micmac*:** l'*Institute National de l'Information Géographique et Forestière* (*IGN France*) ha desenvolupat un software per fer correlació d'imatges multi-escala amb dos aplicacions principals: càlcul de models numèrics d'alçades per correlació i obtenció de lligams fotogramètrics [27]. El codi de Micmac s'ofereix gratuïtament a través de http://logiciels.ign.fr/?Micmac. A [28] es presenta una aplicació per obtenir *DSM*[8] de zones muntanyoses i urbanes a partir de correlació entre imatges *SPOT5* i *Pléiades* respectivament. Aquestes aplicacions utilitzen *Micmac* per dur a terme la correlació i l'ajust.

- ***Eigen*:** és un paquet de llibreries d'àlgebra lineal escrites en C++ i de lliure distribució. Inclou un ampli ventall d'algorismes per operacions matricials, és genèric i amb una interfície molt elegant: la crida a funcions és molt intuïtiva, quasi com si es fes anar un programa de càlcul simbòlic com ara *Maple*. Entre moltes entitats que utilitzen *Eigen* per manegar l'àlgebra lineal cal destacar l'ús de Google per aprenentatge artificial, visió per computador i optimització i, en concret, la llibreria *Ceres* que s'explicarà més endavant.

[8] *DSM (Digital Surface Model)* és un anglicisme per un model numèric d'alçades en què l'alçada inclou arbres, edificis, etc.

- ***GENA* (*Generic Extensible Network Approach*):** és un software desenvolupat a mitges entre l'empresa *Geonumerics* i *l'Institut de Geomàtica* de Castelldefels. Per detalls sobre aquest software es pot consultar la tesi doctoral de la Dra. Marta Blázquez [8]. A l'apèndix 1 titulat *Next generation network adjustments* es fa menció de la gran quantitat de softwares que en l'actualitat s'han dissenyat específicament per àmbits de la geomàtica molt concrets. Aquesta fragmentació d'esforços fa que l'avenç de les tècniques comunes a tots els softwares sigui més lenta. A més, aquest fet també dificulta la integració de sensors en un mateix paquet. Les sigles *G* i *E* de *GENA* fan referència a les dues propietats que haurien de tenir els softwares moderns d'ajust de xarxes, i.e. genèric i flexible respectivament. *GENA* pretén abordar una visió genèrica de l'ajust de xarxes integrant observacions de multitud de sensors alhora que permet una futura extensió i adaptació del software.

  Es tracta d'un paquet comercial, però les idees que es plantegen a [9] seran de gran utilitat de cara a implementar el nou software.

- ***Carmen*:** aquest programa tal i com es descriu a [29] és un dels primers esforços per crear un software modular per tal que sigui extensible. Permet considerar tipus de càmeres molt genèriques.

- ***Bundler:*** és un software escrit en C/C++ que permet obtenir una reconstrucció tridimensional d'una escena a partir d'una col·lecció d'imatges sense posició ni orientació aproximades. Tampoc requereix que estiguin ordenades. A diferència de la majoria de softwares presentats en aquesta recopilació, a més de fer l'ajust de feixos, fa correlació automàtica i calcula aproximacions inicials. El nucli d'ajust de feixos que utilitzen és una variant del paquet *SBA* de *Lourakis* i *Argyros* [3] [23].

  Al voltant d'aquest software han crescut un gran nombre de projectes com ara *Modeling the world from Internet photo collections* [30], *Photo tourism: exploring image collections in 3D* [31] i *Building Rome in a day* [11], entre molts altres. Així doncs, es tracta d'un bon exemple d'aplicació de *SFM* a la fotogrametria arquitectònica des del punt de vista de la visió per computador.

- ***ACX*:** és el software que fa servir actualment l'*ICGC* als seus departaments de producció i desenvolupament. Els seus punts més forts són l'estabilitat i el control estadístic rigorós. Els punts febles són la manca de robustesa (solucionable implementant-hi mètodes robustos), l'eficiència (caldria fer un tractament més especialitzat de matrius disperses), i principalment la manca de flexibilitat. S'explica en detall a l'apartat: *3 Anàlisi del software vigent a l'ICGC*.

- ***Ceres Solver*:** és un paquet de llibreries molt genèric que ha publicat *Google*. S'ha escollit *Ceres Solver* com a motor del nou mòdul perquè té totes les propietats descrites a l'inici d'aquest apartat. El principal avantatge és que es tracta d'un nucli d'ajust molt genèric que té suport continuat gràcies a la retroalimentació d'usuaris d'arreu del món. A més *Ceres* es recolza en llibreries especialitzades, com ara *Eigen*, que resolen problemes concrets de forma molt eficient i que al seu torn s'actualitzen de forma independent.

  Les idees i consells que es donen a [9] seran de gran utilitat de cara a fer que el nou mòdul *Ceres* sigui flexible. L'ús d'aquest paquets s'explica en detall a l'apartat *4 Implementació del nou software.*

3 Part II. Anàlisi del *software* vigent a l'ICGC

3.1 Punt de vista teòric

Abans de començar a implementar el nou mòdul, s'analitzarà el software vigent ja que més endavant s'utilitzarà com a referència per comparar els resultats obtinguts. Per començar es farà una breu descripció teòrica de com funciona *ACX*. Com va dir *Kurt Lewin* l'any 1952, "No hi ha res més pràctic que una bona teoria".

3.1.1 *GEOTEX - ACX*

Tal i com es defineix a [32], *GeoTeX* és un sistema per a la determinació de punts capaç de considerar qualsevol tipus de model geomètric funcional en els àmbits de la geodèsia i la fotogrametria. *GeoTeX* es va començar a desenvolupar a finals de 1988 amb l'objectiu de servir les necessitats d'un entorn de producció i desenvolupament al mateix temps. En aquest sentit, cal arribar a un compromís entre les necessitats d'ambdós departaments, però amb l'avantatge d'un únic software que permet unir esforços. Segons [32] *GeoTeX* s'ha dissenyat en base a tres principis: genèric (aplicable als àmbits de geodèsia i fotogrametria en els entorns de desenvolupament i producció alhora), adaptable (permetent implementar nous models) i portable (compatible amb diferents plataformes). *ACX* és el nom del nucli del sistema *GeoTeX* i és el mòdul encarregat de fer l'ajust d'observacions pel mètode general de mínims quadrats. *GeoTeX* es pot considerar com la plataforma que engloba *ACX* i totes les llibreries, definicions de formats, definicions de tipus de dades, sistemes de referència, canvis de dàtum, interfícies d'usuari, eines auxiliars per transformacions de coordenades, representació gràfica i un llarg etcètera. Tot el sistema *GeoTeX* es basa en un únic format d'arxiu donat pel descriptor de formats *AdIL*.

Els fitxers d'entrada a *ACX* representen els objectes següents

- Observacions: incloent en un mateix arxiu observacions fotogramètriques, de recolzament, trajectòria *GPS/INS*, offset des de l'antena *GPS* fins a certa referència solidària amb la càmera, línies de base *GPS* entre vèrtex geodèsics, etc.
- Paràmetres: aproximacions inicials als paràmetres, incloent coordenades de punts a l'espai objecte, orientacions externes, derives lineals i angulars, matriu de desalineament del sensor, paràmetres d'autocalibratge, etc.
- Sensors: s'inclouen les dades de calibratge de qualsevol nombre càmeres fotogramètriques. *ACX* consultarà només aquelles càmeres que necessiti en funció de les dades d'entrada. Aquestes dades es consideraran com a fixes.
- Opcions: la parametrització de l'ajust s'inclou en un arxiu que serà llegit per *ACX*. Això permet evitar que l'usuari hagi d'entrar les opcions de l'ajust en temps d'execució a través d'una interfície d'usuari.

En els següents apartats primer analitzarem els models i després el mètode de resolució.

3.1.2 Models funcionals i estocàstics d'*ACX*

ACX considera una alternativa al mètode general dels mínims quadrats que consisteix en redefinir els models de manera que sigui possible aïllar cada observació i expressar-la únicament en funció de paràmetres. Aquesta alternativa permet adoptar el mètode de les equacions d'observació per resoldre un ajust en què les fotocoordenades no són les úniques observacions que participen a les equacions de col·linealitat.

Les observacions que intervenen en l'ajust de feixos provenen d'ajustos previs. Per exemple, els punts de recolzament s'observen a camp mitjançant tècniques de *GPS* diferencial respecte d'estacions de referència i aquestes al seu torn han estat ajustades dins d'una xarxa geodèsica. Segons [1] incorporar les correlacions entre observacions a la matriu de covariància suposa un cost computacional massa gran. És per aquest motiu que *ACX* no considera correlació entre observacions i, per tant, és possible adoptar l'alternativa proposada.

La clau consisteix en afegir més models a l'ajust. Es tracta de definir un model específic per cada tipus d'observació seguint el criteri següent:

- Al model d'equacions de col·linealitat només es consideren com a observacions les fotocoordenades. La resta de variables, siguin observades o no, tindran consideració només de paràmetres.
- Es crearà un nou model per cada una de les observacions implicades en les equacions de col·linealitat (recolzament, *GPS*, *offset*...).

Per exemple, tornant al cas de les observacions de recolzament l'alternativa consisteix en crear els dos models següents:

- Model de col·linealitat: on només les fotocoordenades seran considerades observacions. Les coordenades de tots els punts, tant punts de control menor com de recolzament tindran només la consideració de paràmetres.
- Model d'observacions de recolzament: aquí definim la relació entre les coordenades observades i les coordenades paràmetre. El cas més simple seria considerar el model identitat, observació igual a paràmetre.

En quant a cost de computació, la diferència entre aquest mètode i el mètode general dels mínims quadrats ve donada per un canvi en el nombre d'equacions i d'incògnites. El mètode alternatiu requerirà incorporar més equacions ja que cal definir nous models. En canvi, el mètode general incorpora els paràmetres addicionals que s'afegeixen en considerar la jacobiana respecte de les observacions.

Però no sempre serà possible adoptar aquesta alternativa. Per exemple, en el cas de voler ajustar imatges *SPOT* el mètode implicaria considerar dues variables auxiliars que es corresponen amb les observacions píxel fixades i que s'actualitzen a cada nova iteració.

3.2 Punt de vista pràctic

Un cop coneguda la teoria sobre els models que té implementats l'*ACX* i abans de crear un nou software s'ha considerat convenient emular el comportament d'*ACX* per veure si "fa el que la teoria diu que fa".

Es duran a terme dues emulacions i es contrastarà el seu comportament respecte del d'*ACX*. Uns resultats positius no garantiran ni demostraran res. Simplement, en cas d'haver comès alguna equivocació a l'hora d'implementar el model matemàtic, es posarà de manifest que la teoria exposada a l'apartat anterior no és correcta. En paraules de *Richard Feynman*, "*It doesn't matter how beautiful your theory is, it doesn't matter how smart you are. If it doesn't agree with experiment, it's wrong*".

3.2.1 Fulls de càlcul

Treballar amb un full de càlcul permet implementar un ajust de pocs fotogrames i obtenir resultats ràpids sense haver de preocupar-se sobre els mètodes numèrics. Les rutines per operacions matricials que tenen integrades els full de càlcul no són molt eficients, però el fet de poder

representar el contingut de les variables en taules permet una visió gràfica de l'estructura de les matrius del sistema. A més, els blocs petits comporten poques operacions i llavors es considerarà que els errors d'arrodoniment que pugui tenir el full de càlcul són menyspreables.

L'emulador desenvolupat consisteix en un full de càlcul de *Microsoft Excel* anomenat *emulacx.xlsx*. S'han resolt dos problemes. El primer considera un sol model estereoscòpic i servirà per mostrar la visualització de l'estructura de les matrius del sistema. El segon problema considera 30 imatges (3 passades i 10 imatges per passada) i servirà per determinar diferències en l'espai de paràmetres respecte d'*ACX*. En ambdós s'han considerat com a paràmetres punts objecte i orientacions i com a observacions les fotocoordenades i els punts de suport. S'ha afegit soroll gaussià tant a les observacions fotogramètriques (2 μm) com a les observacions de punts de recolzament (4 cm).

A la figura 3.1 es mostra l'estructura de la matriu de disseny. Es ressalten els elements no nuls de les derivades de les equacions de col·linealitat respecte dels paràmetres de punts objecte (caixes verdes) i respecte dels paràmetres d'orientació externa (caixes vermelles). Les caixes grogues corresponen a les matrius identitat que relacionen observacions de recolzament amb punts objecte. Els elements nuls es mostren en blanc.

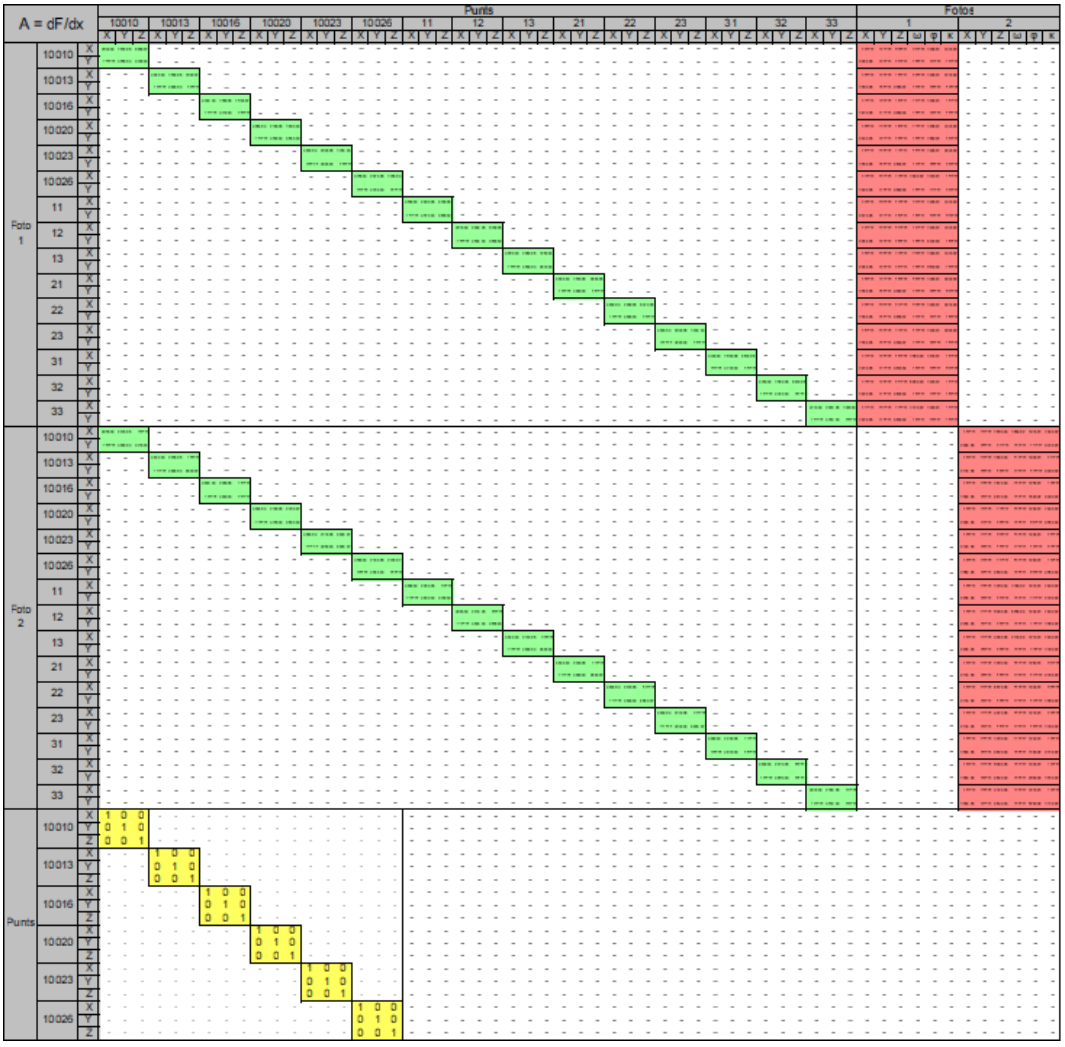


Figura 3.1

Com es pot comprovar en base a la Figura 3.1, a les emulacions només s'han considerat els models definits per les equacions de col·linealitat i per les equacions identitat corresponents a les observacions de punts de recolzament. En cas d'afegir altres models a l'emulador caldria afegir més files i columnes [1]. Per exemple:

- Paràmetres d'autocalibratge: a la dreta de les caixes vermelles caldria afegir tantes columnes com paràmetres d'autocalibratge. Aquestes caixes serien totalment plenes ja que els paràmetres intervenen en totes les imatges i per totes les observacions fotogramètriques.
- Observacions *GPS*: caldrà afegir tantes files com observacions *GPS* de la mateixa manera que els grups de caixes grogues d'observacions de punts de recolzament, però sota les columnes de paràmetres d'orientació externa. Sense considerar un *offset*, l'observació *GPS* es pot considerar com una observació directa del centre de projecció.
- *Offset* d'antena: si es considera un *offset* d'antena només a mode de paràmetres, llavors caldrà afegir tres columnes més que tindran valors de derivada no nuls per les fileres d'observacions *GPS*. Si a més es volen considerar mesures per aquests valors d*'offset*, caldrà afegir unes fileres amb caixes identitat com les grogues, però sota els paràmetres d'*offset*.

3.2.2 Programes de càlcul simbòlic

Amb programes de càlcul simbòlic com *Maple, Maxima, Mathematica, Scilab...* també es podria haver emulat *ACX* obtenint visions gràfiques més sofisticades i molt probablement millorant l'eficiència. No s'ha considerat necessari tenint en compte que l'emulació és només un pas previ a la implementació del software. En comptes d'això s'han emprat aquests programes, en concret *Maple i Maxima*, únicament per:

1. Fer el producte de les matrius de rotacions de forma simbòlica:

$$R_{\omega\varphi\kappa} = R_\kappa R_\varphi R_\omega$$

$$R_{\omega\varphi\kappa} = \begin{pmatrix} \cos\kappa & -\sin\kappa & 0 \\ \sin\kappa & \cos\kappa & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\varphi & 0 & \sin\varphi \\ 0 & 1 & 0 \\ -\sin\varphi & 0 & \cos\varphi \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\omega & -\sin\omega \\ 0 & \sin\omega & \cos\omega \end{pmatrix}$$

2. Calcular la jacobiana de la matriu de disseny de forma analítica aprofitant les potents eines de diferenciació simbòlica. En concret, *Maple 17* té una funció anomenada *jacobian* a la que cal passar-li les funcions i les variables a diferenciar. Un cop obtinguda la jacobiana, s'ha fet una primera simplificació de les expressions amb la funció *simplify* i una última simplificació manual[9] en aquells casos que el resultat de *simplify* no ha estat òptim.

3.2.3 Emulació amb *Python*

Paral·lelament, el Joan Arnaldich va desenvolupar un altre emulador d'*ACX* programant en *Python* 2.7, un llenguatge de programació àgil i potent. A diferència de llenguatges de programació com C/C++, *Python* no compila directament a codi màquina creant un executable *.exe*. En comptes d'això, fa una pseudo-compilació o interpretació que tradueix a un codi específic *.pyc* que serà

[9] Per fer la simplificació manual s'han adoptat les variables auxiliars considerades a [33].

processat per una màquina virtual de *Python* [34]. A continuació es descriu breument algunes característiques de l'emulador resultant, *aeropy.py*:

- Format de dades *YAML*: la lectura de les dades que entren a l'ajust (observacions i aproximacions inicials) es duu a terme interpretant els fitxers en format *YAML*. *YAML* és un estàndard per format de dades semblant a *XML*, però segons els autors, no ho és: "*YAML ain't Markup Language"* [35]. Escriure els fitxers en aquest format ha permès simplificar la lectura de dades d'*aeropy* de cara a obtenir resultats ràpidament. Un exemple de l'estructura del fitxer d'entrada que ha dissenyat el Joan:

```
observations:
  - observation_model: collinearity_observation
    # OJU!!! Micròmetres
    data: |
      1     10010     39.80         75709.60      2.0     2.0
      1     10013     12.19         295.13        2.0     2.0
      ...
  - observation_model: identity_observation
    data: |
      10010 40027.176 4633578.149 655.862 0.04 0.04 0.04
      10013 40032.183 4632312.283 627.757 0.04 0.04 0.04
      ...
```

- Ús de llibreries especialitzades: s'ha aprofitat el potencial de llibreries de codi lliure com *numpy* i *scipy* per no haver de preocupar-se pels problemes de càlcul numèric i àlgebra lineal. Per exemple *scipy* conté algorismes, mètodes i operadors sobrecarregats que permeten resoldre el sistema lineal en poques línies de codi. Per exemple, *spsolve* resol un sistema lineal tenint en compte l'estructura dispersa de les matrius:

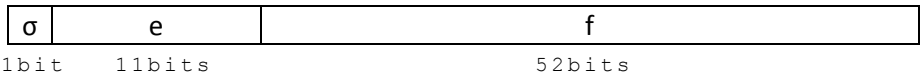
```
T = At*W*U
X = spsolve(At*W*A, T)
```

3.2.4 Posant a prova el model matemàtic

La implementació dels dos emuladors anteriors ha servit per detectar possibles equivocacions d'implementació en el model matemàtic, en l'estratègia de resolució del sistema no lineal i en el mètode de resolució del sistema lineal. Amb aquesta finalitat s'ha pres com a referència *ACX*.

Les discrepàncies entre software i emulador vindran donades per tres fonts:

- Equivocacions conceptuals: degudes a les suposicions sobre els models i els mètodes de resolució adoptats per *ACX*.
- Equivocacions d'implementació de l'emulador: suposant que un software extensament validat com *ACX* no té errors d'implementació, una discrepància significativa respecte d'*ACX* serà deguda probablement a un error d'implementació de l'emulador.
- Errors d'arrodoniment: un cop descartades les dues fonts anteriors, les discrepàncies residuals haurien de ser degudes a la diferència en la representació numèrica. És a dir, el nombre de xifres significatives que es conserven per una variable de tipus real. A tall d'exemple s'il·lustra la informació que conserva una variable punt flotant de doble precisió en els 64 bits d'espai que té reservats [36]:



On σ és el signe ($-1^0 = 1$ o $-1^1 = -1$), $e \in [0,11]$ és l'exponent de 2^e i f és la part decimal representada per un nombre binari de 52 bits

A mesura que es fan operacions de "punt flotant" amb nombres binaris es perd la informació de la part no representable. Un anàlisi rigorós de la propagació d'errors està fora de l'abast d'aquest treball.

A continuació s'exposen els criteris per avaluar les diferències d'implementació entre els dos emuladors i *ACX*:

1. ***Misclosures***: és el nom amb el que es coneixen en la literatura anglosaxona als residus previs a l'ajust, és a dir, les diferències entre les observacions i els valors de les mateixes observacions calculades a partir de les aproximacions inicials. En cas de detectar diferències en el vector de *misclosures* voldria dir que existeix alguna equivocació a l'hora de definir el model matemàtic.
2. **Diferències en el valor del *pas* calculat**: comparar el vector de correccions als paràmetres per a cada iteració permet detectar possibles diferències en l'estratègia adoptada per resoldre el sistema no lineal. Tant els emuladors com *ACX* fan servir l'estratègia de *Gauss-Newton*, així que no s'espera trobar diferències en el valor del *pas*.
3. **Exactitud del resultat**: s'ha calculat la diferència entre els paràmetres ajustats (coordenades de punts i orientacions) d'*ACX* i dels dos emuladors. Si es detecten diferències aquí no serà possible saber l'origen de les equivocacions (poden provenir de qualsevol punt del procés d'ajust). Però en el cas d'obtenir resultats amb discrepàncies degudes només a errors d'arrodoniment, es pot donar per vàlid l'experiment sencer.

4 Part III. Implementació del nou mòdul

4.1 *Ceres Solver*

A principis de 2010 *Google* va desenvolupar *Ceres Solver*, una llibreria de C++ que permet resoldre sistemes no lineals pel mètode general dels mínims quadrats. Des de llavors l'empresa fa ús d'aquest software per algunes de les seves pròpies aplicacions [37]. Per exemple, determinar l'orientació de les càmeres utilitzades a *StreetView*, crear els *3D Photo Tours* per *Google Maps*, formar panoràmiques i com a motor d'ajust de feixos del *Tango Project*. Els més de tres anys d'ús intensiu del software els ha permès obtenir un codi força depurat i extensament validat. El maig de 2012 *Google* va començar a distribuir *Ceres Solver* com a codi lliure amb suport per *Linux*, *OSX (Mac)* i *Windows* amb *Visual Studio*.

4.1.1 Documentació

4.1.1.1 El manual de Ceres

Els autors de *Ceres Solver*, *Sameer Agarwal* i *Keir Mierle*, van publicar el novembre de 2012 un manual escrit i compilat en *LaTeX* que ha servit de guia per la major part d'aquest treball [2]. Així i tot, els mateixos autors adverteixen que el manual no s'actualitza amb freqüència i recomanen consultar el codi font com a referència més actual.

El manual conté tres seccions:

- *Building*: es tracta del manual d'instal·lació de *Ceres*. S'explicarà més endavant a l'apartat *4.1.2 Compilació i enllaç amb llibreries Ceres*.
- *Tutorial*: a través d'una sèrie de problemes d'optimització per mínims quadrats s'explica el funcionament de *Ceres* i alhora li serveixen a l'usuari com a punt de partida per començar a implementar el seu problema. Com la majoria de manuals de programació, comença amb una aplicació simple tipus *Hello World*! En aquest cas, *Hello World!* és una aplicació de consola escrita en C++ que permet resoldre un problema trivial, trobar el valor de *x* que minimitzi la funció $\frac{1}{2}(10-x)^2$. En base a aquest problema s'explica què és i com implementar un *functor*, un *cost function*[10], etc i quins altres objectes cal crear abans de resoldre el problema. El *tutorial* acaba amb un exemple de programa més sofisticat (i més útil de cara a aquest projecte), l'ajust de feixos. El codi d'aquest exemple el publiquen amb el nom *bundle_adjuster.cc*.
- *Reference*: aquí s'explica com funcionen les classes de *Ceres* i la teoria que hi ha al darrere. Per exemple, s'explica quins objectes cal crear per tal de resoldre un problema de mínims quadrats amb *Ceres* (diferents tipus de *cost functions*, *loss functions*, *parameter blocks*, *residual blocks*, *problem*). També es fa una breu descripció dels mètodes de resolució per tal que l'usuari sàpiga quines són les opcions que més li convenen pel seu problema.

[10] En aquest treball es distingeix entre el concepte genèric "funció de cost" per diferenciar-la de la implementació concreta a *Ceres Solver* denominada "*cost function*".

4.1.1.2 El codi font com a manual d'usuari

La documentació més actualitzada es troba a les seccions comentades dins de les capçaleres del codi font, és a dir, als arxius *.h* i *.cc* de les classes de *Ceres*. Per moltes classes un gruix important dels arxius correspon a les seccions comentades amb explicacions detallades de com utilitzar-lo, exemples d'ús i, fins i tot, nocions teòriques bàsiques de les matemàtiques que hi ha al darrere del codi. Per exemple, de les 422 línies de codi de *covariance.h*, 274 són comentaris i de les 667 línies de codi de *jet.h* (implementació de nombres duals per la diferenciació automàtica), 306 són comentaris.

4.1.1.3 Google groups

Els mateixos desenvolupadors moderen un fòrum de *Google Groups* on els usuaris poden preguntar sobre el funcionament de la *Ceres*. Per una banda, els usuaris tenen el privilegi de demanar els seus dubtes als mateixos desenvolupadors. Per l'altra, ells es beneficien de la retroalimentació dels usuaris de cara a resoldre *bugs* i rebre suggeriments de millora.

Un exemple d'aquesta realimentació positiva és la implementació de la classe continguda a *covariance.h* [38] arrel del suggeriment d'un usuari. L'usuari presenta a través de *Google Groups* un mètode per obtenir submatrius de la matriu de variància-covariància sense haver d'invertir la matriu normal sencera. En resposta, els creadors de *Ceres* van desenvolupar el maig de 2013 una nova llibreria de *Ceres* per calcular la matriu de variància-covariància.

De cara a aquest TFG, en cert moment va sorgir el dubte de com incorporar desviacions a priori a les observacions de l'ajust de feixos. Com que al manual oficial de *Ceres* no hi havia cap informació al respecte vam cercar al *Google Groups* de *Ceres*. Algú ja havia formulat la pregunta i publicat sota el títol: *Weighting / Covariances - Google Grups* [39]. La pregunta la formula un usuari, un tal *Don Venable*:

Do you have any recommendations on either how to use Ceres to incorporate prior information through weighting, or can you comment on a path forward for Ceres support in handling prior statistics into the bundle adjustment?

Sameer Agarwaal, un dels desenvolupadors, contesta amb tres propostes.

1. Escalar els residus directament dins de la *cost function*
2. Fer servir la classe *ConditionedCostFunction* de *Ceres*. Segons diuen els desenvolupadors, amb aquesta proposta no és possible tenir en compte la correlació entre observacions.
3. Fer servir la classe *ScaledLoss* també de *Ceres*. Amb aquest mètode estaríem associant la mateixa desviació a totes les observacions del mateix tipus. No ens interessa.

En una sèrie de preguntes i respostes es proposen alternatives com ara ampliar la llibreria *Ceres* amb una nova subclasse de *CostFunction* per respondre a la demanda d'usuaris. Es tractaria d'una *WeightedCostFunction* (filla de *CostFunction*) que considerés els pesos de les observacions.

Després de barallar-me durant dues setmanes intentant adaptar alguna de les solucions proposades i veient que no aconseguia resoldre el meu cas vaig decidir intervenir al fòrum amb la següent pregunta:

Hi Sameer,

I am currently using ScaledLoss to apply different weights to different residual blocks in a bundle adjustment and it works just fine. However, I am now considering upgrading to methods 1

or 2 (as referred to in your reply to Don last year) because I need different weights within each residual block, i.e. one scalar per residual.

Hard-coding the weights into the CostFunction object is not an option since I want weights to be user's input. Considering that I need automatic differentiation, but unlike Paul, I do NOT need to account for correlation between residuals, which method would you suggest? Using ConditionedCostFunction or perhaps creating my own CovariaceMultiplier function? The WeightedCostFunction would come in so handy now…

Also, I checked ConditionedCostFunction documentation and I am having a little trouble defining the CostFunctions that should be passed into conditioners vector. Assuming that I just need the conditioner to return "residual times scalar", is it enough to define a SizedCostFunction<1,1> where the residual of the wrapped cost function is passed to the conditioner CostFunction as a parameter? Unfortunately, I am not able to find a full example of an implemented ConditionedCostFunction in Ceres tutorial & reference manual or elsewhere on the web.

La resposta d'en *Sameer Agarwal* em va fer obrir els ulls. Em va contestar que només cal modificar la crida a la *cost function* passant les desviacions per paràmetre. Un cop a dins de la *cost function* es poden escalar els residus considerant un pes $p_i = \frac{1}{\sigma_i}$.

4.1.2 Compilació i enllaç amb llibreries *Ceres*

En tractar-se d'un paquet de llibreries, la instal·lació es coneix com a "construir" el projecte i fer que compili sota un determinat sistema operatiu. Tal i com es diu a [2] i [37], la compilació sobre *Windows* és més complicada que sobre *Linux* o *MacCal* ja que no hi ha una manera automatitzada d'instal·lar els paquets. A més, a *Windows* només consideren l'opció de compilar amb *Microsoft Visual Studio* 2010 o més superior. Cal agrair especialment el gran esforç i dedicació del Joan Arnaldich i Bernal per compilar *Ceres* sobre *Windows* en base a dos compiladors *cygwin* i *mingw* i sense disposar de cap versió de *Visual Studio*.

- *Cygwin*: va ser la primera compilació sobre la que es va treballar i desenvolupar gran part del codi. L'inconvenient és que no va ser possible enllaçar les llibreries Ceres amb *SuiteSparse*, una dependència que permet un tractament avançat de matrius disperses.
- *Mingw*: va ser una segona compilació més complicada que la primera que sí va permetre enllaçar amb *Ceres* amb *SuiteSparse* accelerant la resolució dels blocs fotogramètrics més grans.

Ceres està dissenyat per fer que enllaci amb una sèrie de llibreries de tercers que permeten dur a terme tasques concretes com ara [2]:

- *cmake*: és un instal·lador multi-plataforma que permet "construir" el projecte i que compili.
- *Eigen3:* per operacions matricials bàsiques
- Controlar l'estratègia iterativa del sistema no lineal
- La resolució del sistema lineal
- *CXSparse* i *SuiteSparse*: tractament de matrius disperses bàsic i avançat, respectivament. *SuiteSparse* al seu torn depèn d'altres llibreries com *BLAS* i *LAPACK*.

- *Google-glog i gflags:* detecció d'errors i fer proves per comprovar el correcte funcionament de la compilació de *Ceres*.
- Contribucions d'usuaris: alguns usuaris de *Ceres* també proposen mètodes alternatius i millores. Per exemple, *Markus Moll* aporta nous algorismes per l'estratègia *Dogleg*, suport per *SuiteSparse4* i resol nombrosos "bugs" de programació.

4.2 Treballant amb *Ceres*

Ceres no és un programa, és un paquet de llibreries escrites en C++. Així doncs, per resoldre qualsevol problema cal que l'usuari faci un programa, preferiblement en el mateix llenguatge que les llibreries, i des d'ell cridi les classes definides per *Ceres*. Per aquest treball s'ha escrit una aplicació de consola per Windows 32bit en C++.

4.2.1 Què fa *Ceres* per l'usuari?

Malgrat *Ceres* sigui una llibreria molt complerta, l'usuari és el responsable de crear objectes *Ceres* amb la informació necessària i gestionar després els resultats. Cal que el programa que implementi l'usuari dugui a terme les següents tasques:

1. Carregar observacions i aproximacions inicials dels paràmetres
2. Definir els models funcionals i estocàstics. És responsabilitat de l'usuari saber formular el problema a resoldre en termes de funcions de cost.
3. Assignar pesos a les observacions en funció de les desviacions associades
4. Definir funcions de mètodes robustos si l'usuari així ho vol
5. Escollir les opcions generals de resolució del sistema no lineal (criteris de convergència, mètode de resolució del sistema lineal, etc.). És aquí on entra en joc el coneixement teòric de l'usuari sobre el problema concret que vol resoldre.
6. Crear objectes que necessita *Ceres* com a entrada
7. Consultar o calcular informació estocàstica que *Ceres* no reporta per defecte
8. Imprimir els resultats

Per aquest treball, la teoria darrere dels punts 2 i 5 anteriors s'explica a la secció *Part I. Recerca*, als apartats *2.2* a *2.3* i *2.4* a *2.6* respectivament.

Ceres s'encarregarà de:

1. Linealitzar el sistema (en cas d'escollir diferenciació automàtica)
2. Muntar les matrius de disseny i termes independents del sistema lineal
3. La reordenació de files i tractament de matrius disperses
4. Aplicar la tècnica escollida de mètodes robustos
5. Gestionar l'estratègia de resolució del sistema no lineal
6. Resoldre els sistemes lineals corresponents a cada iteració
7. Controlar la convergència

4.2.2 *Cost functions* i *functors*

Les *Cost functions* són classes de *Ceres* que permeten definir els models funcional i estocàstic.

A la pràctica, un objecte *cost function* pren un vector de paràmetres com a entrada i retorna un vector de residus i la jacobiana. El càlcul de la jacobiana es duu a terme amb tècniques de diferenciació automàtica, diferenciació numèrica o bé manualment especificant explícitament les derivades al cos de la *cost function*. Els *functors* són funcions definides per l'usuari que s'encarreguen de la tasca principal de les *cost funcions*, i.e. calcular els residus donades les observacions i els paràmetres relacionats. Els *functors* són útils quan es defineixen *cost functions* que utilitzen diferenciació automàtica.

A continuació es presenten diferents tipus de *cost functions* que es poden crear:

- *Simple cost function*: és la base d'altres *cost functions*. L'usuari ha d'especificar les dimensions dels vectors de paràmetres i el nombre de residus d'entrada i també implementar-hi un mètode *Evaluate()* per calcular els residus i, opcionalment, la jacobiana.
- *Sized cost function*: és un cas particular de l'anterior en què s'aprofita el fet de conèixer la mida dels vectors de paràmetres i el nombre de residus d'entrada en temps de compilació i així poder crear una *cost function* amb les mides ja definides. L'usuari només haurà d'implementar el mètode *Evaluate()*.
- *NumericDiff cost function*: implementa el mètode de les diferències finites per aproximar la derivada de la funció de cost. Es pot escollir el mètode de *forward difference* per avaluar l'increment com $f(x+h)-f(x)$ o bé el mètode de *central difference* que avalua l'increment centrat en *x*, i.e. $f(x+h)-f(x-h)$. El mètode *central difference* és més precís ja que calcula la derivada centrada en el paràmetre *x*. L'inconvenient és que triga més ja que *forward difference* aprofita que $f(x)$ s'ha de calculat prèviament.
- *AutoDiff cost function*: per tal d'aprofitar el potencial de la diferenciació automàtica, caldrà que l'usuari creï una nova classe (e.g. *MyScalarCF*) que contingui un *functor* per calcular residus de tipus *template* a partir de paràmetres també de tipus *template*. D'aquesta manera, un mateix paràmetre o residu podrà ser tant de tipus *double* com de tipus *Jet* (nombres duals) alhora. Finalment, caldrà crear un objecte de tipus *AutoDiff cost function* que necessita com a entrada les observacions i la nova classe *MyScalarCF*. A [2] es recomana que es faci servir *AutoDiff cost function* sempre que sigui possible.

A mode d'exemple es mostra el disseny de la *AutoDiff cost function* que defineix el model funcional de les observacions de punts de control d'aquest projecte. Per motius de simplicitat, s'ha fet una abstracció en un pseudocodi molt semblant a C++.

En l'exemple la nova classe *MyScalarCF* es diu *CObsRecolzCF*. Es calculen residus *res[i]* per les tres components *i = 1..3* del punt objecte amb paràmetres *par[i]* i amb magnituds observades *obs[i]* de desviacions especificades per *dev[i]*.

```
class CPuntSuportObsCF
{
    CObsRecolzCF(double obs[0], obs[1], obs[2],
                              dev[0], dev[1], dev[2]) {...}

    template<typename T>
    bool operator()(const T* par, T* res) const
    {
```

```
        // Residus de punts de recolz escalats per la desviació
        for (int i=0; i<3; i++)
          res[i] = ((obs[i]) - par[i])/dev[i] ;

        return true;
      }

    private:
      double obs[3], dev[3];
};
```

Finalment s'ha creat l'objecte *AutoDiff cost function* passant-li les observacions i un objecte de la nova classe *CObsRecolzCF*:

```
Ceres::CostFunction* cost_function =
  new Ceres::AutoDiffCostFunction< CObsRecolzCF, 3, 3>(
    new CObsRecolzCF(GCP_obs_X, GCP_obs_Y, GCP_obs_Z,
                     GCP_dev_X, GCP_dev_Y, GCP_dev_Z
    )
  );
```

4.2.3 Parameter blocks i residual blocks

Ceres treballa internament amb paràmetres i observacions agrupades per tal d'optimitzar l'estructura de la matriu de disseny i minimitzar el nombre d'índex necessaris per accedir als elements.

És responsabilitat de l'usuari definir quins paràmetres pertanyen a cada *parameter block* i quines observacions pertanyen a cada *residual block*. Per fer-ho cal tenir en compte la relació entre paràmetres i residus. És a dir, quins paràmetres cal passar a una funció de cost per que pugui calcular un determinat residu. Una bona definició de *parameter blocks* i *residual blocks* facilitarà que *Ceres* optimitzi la dispersió de la matriu de disseny.

En general, els *parameter blocks* de *Ceres* es correspondran amb els *parameter groups* d'*ACX*. A més, en cas que en el model hi hagi una correspondència entre observacions i residus llavors els *residual blocks* de *Ceres* també es correspondran amb els *observation groups* d'*ACX*.

4.2.3.1 Parameter blocks

En general un *parameter block* agrupa aquells paràmetres que pertanyen a una mateixa entitat del problema. A l'apartat de *FAQs* del manual de *Ceres* s'exposen els criteris per definir els *parameter blocks*. En concret, per saber si cal agrupar un conjunt de paràmetres en un mateix *parameter block* és suficient si ens fem la següent pregunta: existeix alguna observació en el model que depengui d'un subconjunt d'aquests paràmetres? Si la resposta és no, llavors els podem agrupar.

Aquests són els *parameter blocks* que s'han definit en aquest projecte:

1. Les coordenades d'un punt a l'espai objecte: *X*, *Y*, *H*
2. Els 6 paràmetres d'orientació externa d'una imatge
3. Els 12+3 paràmetres d'autocalibració d'*ebner*
4. Les 3 components de l'*offset*

4.2.3.2 Residual blocks

Aquests són els *residual blocks* que s'han definit en aquest projecte:

1. Un vector residu fotogramètric: *x, y*
2. Un vector residu de mesura *GPS*: *X, Y, H*
3. Un vector residu de mesura d'*offset* d'antena *GPS*: *dX, dY, dZ*
4. Un vector de residus amb les 12+3 pseudoobservacions de paràmetres d'autocalibració

4.2.4 Loss funcions

A l'entorn de *Ceres*, *Loss functions* és el nom que reben aquelles funcions que permeten implementar els mètodes robustos. Una *loss function* "envolta" una *cost function* per tal de modificar els residus d'aquelles observacions que poden estar afectades per errors grollers. Si es considera l'optimització d'una funció qualsevol, una *loss function* pren la forma genèrica:

$$\frac{1}{2}\sum_{i=1}^{n}\rho(\| f_i(x_{i1},\ldots,x_{ik})\|^2)$$

on ρ és la *loss function*, f és la *cost function*, x_i són els paràmetres implicats en la funció de cost f_i, n és el nombre de funcions de cost f que intervenen a l'ajust i k és el nombre de grups de paràmetres.

En cas de que $f_i(x_{i1},\ldots,x_{ik})$ sigui la suma quadràtica de residus i ρ sigui la funció identitat, llavors s'obté el mètode mínim-quadràtic convencional.

Ceres disposa d'un seguit de *loss functions* ja implementades com ara:

- *Null loss o trivial loss*: és la funció identitat i per tant la funció de cost no es modifica:

$$\rho(f(x)) = f(x)$$

- *Scaled loss*: permet multiplicar la funció de cost per un escalar *a*. Cal tenir en compte que s'escalaran totes les observacions d'un mateix tipus pel mateix valor. És a dir, no es distingirà entre dos residus obtinguts amb la mateixa *cost function*.

$$\rho(f(x)) = a\, f(x)$$

- *Huber loss:* és una funció definida a trossos. Es modifica la funció de cost per valors del cost superiors a la unitat:

$$\rho(f(x)) = \begin{cases} f(x), & f(x) < 1 \\ 2\sqrt{f(x)} - 1, & f(x) \geq 1 \end{cases}$$

- *Soft L1 loss*: similar a la *Huber loss* però suavitzada (sense la discontinuïtat deguda a la definició a trossos):

$$\rho(f(x)) = 2\sqrt{1 + f(x)} - 1$$

- *Cauchy loss*: inspirada en la distribució de Cauchy

$$\rho(f(x)) = log(1 + f(x))$$

- *Arctan loss*: s'introdueix un paràmetre *a* en una funció arctangent que determinarà el punt a partir del qual l'observació serà rebutjada com a error groller:

$$\rho(f(x)) = a\tan\left(\frac{f(x)}{a}\right)$$

- Composició de *Loss functions*: s'aplica una segona *cost function* γ al cost resultant d'aplicar una primera *cost function* ρ

$$\gamma(\rho(f(x)))$$

De cara a implementar una nova *loss function* personalitzada a [2] es recomana que compleixi les següents restriccions:

$$\rho(f(x)) = 0$$
$$\rho(f(x)) = 1$$

Si $f(x)$ manifesta errors grollers en les observacions, llavors també:

$$\rho'(f(x)) < 1$$
$$\rho''(f(x)) < 0$$

4.3 El nou mòdul

Un cop coneguda la teoria de com utilitzar les llibreries *Ceres* per resoldre un problema genèric de mínims quadrats, cal aplicar-ho al nou mòdul. Potser l'opció més senzilla hagués estat aprofitar el codi d'un dels exemples del tutorial de *Ceres*, *simple_bundle_adjuster.cc* o bé l'exemple més complert *bundle_adjuster.cc*. En comptes d'aprofitar-los, s'ha optat per començar un mòdul de nou per dos motius:

1. Comprendre i tenir controlat a més baix nivell el funcionament del nou mòdul. El fet de començar de zero i cridar les llibreries a mesura que es necessiten evita que existeixin blocs de codi al fil del programa principal (*main.cc*) que es desconegui la seva funció. És a dir, que no arribi a sorgir la pregunta: *Per què serveix aquest bucle que triga una eternitat? Ni idea. Només sé que si l'elimino el programa peta*.
2. L'exemple *bundle_adjuster.cc* es basa en el model funcional definit a [30] per *Noah Snavely* amb funcions de cost anomenades *SnavelyReprojectionError*. Aquest model funcional només considera punts i orientacions com a paràmetres i les fotocoordenades com a úniques observacions. Per tant, aprofitar aquest codi implicaria haver d'adaptar-lo a les necessitats de l'*ICGC*, és a dir, considerar observacions de punts de recolzament, *GPS*, etc. Les modificacions del codi porten de nou al motiu número 1.

El nou mòdul consisteix en un programa escrit en C++ amb l'estructura bàsica de la figura 4.1:

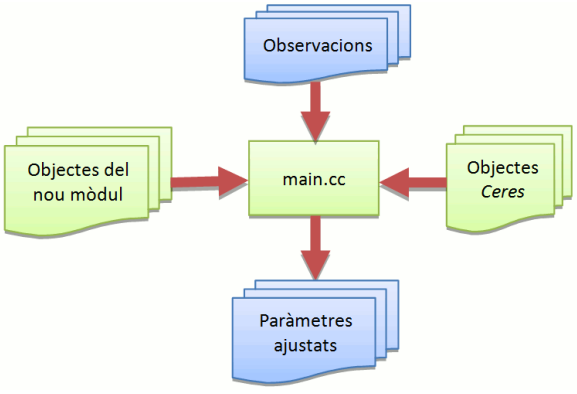


Figura 4.1

4.3.1 Convencions del nou mòdul

Abans de començar a concretar com s'ha implementat el nou mòdul s'exposaran algunes de les convencions que s'adopten. La tria d'aquestes convencions ve donada per un equilibri entre el *modus operandi* de l'*ICGC* i la necessitat de simplificar el codi durant els primers estadis d'implementació.

4.3.1.1 Sistema de coordenades de l'espai imatge

L'origen de coordenades és el centre fiducial i els eixos es defineixen com:

- Eix *x*: direcció definida per la fila central de la imatge. Aproximadament sobre l'eix de vol i creixent segons el sentit de desplaçament de l'avió.
- Eix *y*: direcció definida per la columna central de la imatge. Aproximadament perpendicular a l'eix de vol i creixent cap a l'ala esquerra de l'avió.

4.3.1.2 Sistema de coordenades de l'espai objecte

Per l'espai objecte s'ha considerat un sistema de coordenades cartesià en un espai euclidià i amb un origen arbitrari. Al món real però, els blocs fotogramètrics estan emmarcats en un sistema de referència que ve determinat per la font de les observacions. Així doncs estan lligats directament o indirectament amb un marc de referència. Per exemple, els punts de recolzament que s'han mesurat amb receptors *GPS* estaran indirectament lligats a les estacions de referència i/o a vèrtex geodèsics les coordenades dels quals estan expressades en un determinat sistema de referència. Així doncs, en cert punt caldrà ampliar els models per considerar coordenades geodèsiques, geocèntriques, projecció *UTM*, etc.

4.3.1.3 Correccions de curvatura, refracció i distorsió de lent

Amb l'objectiu de simplificar la implementació del nou mòdul, les correccions de curvatura, refracció i distorsió de lent s'apliquen abans d'entrar a l'ajust de feixos. És a dir, no incorporem desviacions associades a les observacions dels models de correcció ni permetem que es determinin a l'ajust. Les fotocoordenades que entrin al mòdul estaran prèviament corregides.

4.3.1.4 Recolzament aeri

El recolzament aeri consisteix en incorporar a l'aerotriangulació observacions *GPS* i *INS* provinents d'un ajust previ de la trajectòria de l'avió. La relació geomètrica entre la trajectòria de l'avió i l'orientació externa de la imatge en general serà coneguda i vindrà determinada per mesures d'*offset* d'antena, desalineament de la unitat inercial, etc. Un cop ajustada la trajectòria es duu a terme un procés d'interpolació per obtenir les observacions referides al temps d'exposició de la imatge.

Amb l'objectiu de simplificar la implementació del nou mòdul, considerem que la trajectòria ha estat prèviament ajustada i les observacions *GPS* i *INS* interpolades. Per tant, per a cada imatge del bloc tindrem les observacions *GPS* i *INS* corresponents d'aquell instant. La relació geomètrica entre imatge i observació *GPS/INS* vindrà donada per l'*offset*.

4.3.2 Models funcionals

A continuació es defineixen els models funcionals que s'han considerat al mòdul *Ceres*. Tots els models considerats són explícits.

El nou mòdul considera la mateixa alternativa al mètode general de mínims quadrats que *ACX* pels següents motius:

1. Es poden considerar tots els tipus d'observacions que acostumen a intervenir en l'ajust de feixos, és a dir, fotocoordenades, observacions de recolzament, mesures de trajectòria *GPS*, mesura d'*offset* d'antena, etc). Tots ells amb la seva desviació a priori associada.
2. La jacobiana respecte de les observacions és la identitat. Conseqüentment, la matriu normal és més reduïda.
3. No es considera la correlació entre observacions. Al menys per ara.
4. De cara a avaluar l'exactitud del nou mòdul les comparacions amb *ACX* se simplifiquen en cas de considerar el mateix mètode d'ajust.

Per cada un es detallaran les observacions, els paràmetres relacionats i la seva relació en forma d'equacions d'observació o funcions de cost.

Les observacions i paràmetres s'agrupen per blocs d'observacions del tipus $<o_1, o_2 ..., o_m>$ i blocs de paràmetres del tipus $<p_1, p_2 ..., p_n>$ respectivament.

4.3.2.1 Observacions fotogramètriques

És el model funcional principal de l'ajust de feixos definit per la condició de col·linealitat. S'han considerat dues variants d'aquest model, un amb paràmetres d'autocalibratge i un sense.

- Observacions: $<x, y>_{p,i}$ són les fotocoordenades d'un punt *p* mesurat sobre una imatge *i*.
- Paràmetres:
  - Orientació de la imatge: $<X, Y, Z, \ , \varphi, \kappa>_i$ són els 6 paràmetres clàssics d'orientació externa en l'espai objecte. *X, Y, Z* corresponen a les coordenades del punt principal i $, \varphi, \kappa$ són les rotacions respecte dels tres eixos cartesians del sistema de coordenades de l'espai objecte.
  - Punts objecte: $<X, Y, Z>_p$ són les coordenades en l'espai objecte dels elements identificats a les imatges.
  - Paràmetres d'autocalibratge: són els paràmetres que modelen les distorsions de les imatges $<df, dx_0, dy_0, e_{[12]}>_e$
- Constants: $<f, x_0, y_0>_s$ grup de valors donats al certificat de calibratge del sensor *s*. Aquests pseudo-paràmetres s'ajustaran de manera indirecta a través de les correccions corresponents $<df, dx_0, dy_0>_s$ que sí intervindran a mode de paràmetres a l'ajust.
- Model: la relació entre paràmetres i observacions ve determinada per les equacions de col·linealitat.

$$\begin{pmatrix} r \\ s \\ q \end{pmatrix} = R_i^t \left(\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_p - \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_i \right)$$

  - Model sense autocalibratge:

$$\begin{pmatrix} x \\ y \end{pmatrix}_p^i = \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} - \frac{f}{q_p^i} \begin{pmatrix} r \\ s \end{pmatrix}_p^i$$

  - Model amb paràmetres d'autocalibratge:

$$\begin{pmatrix} x' \\ y' \end{pmatrix}_p^i = \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + \begin{pmatrix} dx_0 \\ dy_0 \end{pmatrix} - \frac{f + df}{q_p^i} \begin{pmatrix} r \\ s \end{pmatrix}_p^i$$

$$\begin{pmatrix} x \\ y \end{pmatrix}_p^i = \begin{pmatrix} E_x \begin{pmatrix} x' \\ y' \end{pmatrix}_p^i \\ E_y \begin{pmatrix} x' \\ y' \end{pmatrix}_p^i \end{pmatrix} + \begin{pmatrix} x' \\ y' \end{pmatrix}_p^i$$

On $E_x(x', y')$ i $E_y(x', y')$ són els polinomis *d'Ebner* [40] per la fotocoordenada (x, y).

4.3.2.2 Punts de recolzament

És el model que permet associar observacions de les coordenades de determinats punts de la xarxa fotogramètrica, els punts de recolzament.

- Observacions: $<X, Y, Z>_r$ són les coordenades observades d'un punt de recolzament *r* a l'espai objecte.
- Paràmetres: $<X, Y, Z>_p$ són les coordenades a determinar d'un punt *p* qualsevol a l'espai objecte.
- Model: la relació entre paràmetres i observacions és la identitat.

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_r^{Obs} = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_p^{Par}$$

4.3.2.3 Pseudo-observacions de paràmetres d'autocalibratge

Es tracta d'un model auxiliar que no és necessari per considerar els paràmetres d'autocalibratge, però que permet associar certa desviació a cada paràmetre. Així, és possible fixar o alliberar certs paràmetres a conveniència.

- Observacions: $<df, dx_0, dy_0, e_{[12]}>_e$ són pseudo-observacions en el sentit que, malgrat els valors no provinguin de mesures reals, intervenen a l'ajust a mode d'observacions. Les observacions s'inicialitzen a zero i s'allibera la seva desviació per tal de que es puguin determinar durant l'ajust.
- Paràmetres: $<df, dx_0, dy_0, e_{[12]}>_p$ són els paràmetres que modelen les distorsions de les imatges.
  - *df*: increment en la longitud focal de la càmera
  - dx_0, dy_0: desplaçament del punt principal de simetria
  - $e_{[12]}$: vector amb els 12 coeficients del polinomi d'*Ebner*.
- Model: la relació entre paràmetres i observacions és la identitat.

$$\overbrace{\begin{matrix} df & dx_0 & dy_0 & e_1 & e_2 & \cdots & e_{12} \end{matrix}}^{e} = \overbrace{\begin{matrix} df & dx_0 & dy_0 & e_1 & e_2 & \cdots & e_{12} \end{matrix}}^{p}$$

4.3.2.4 Observacions GPS

En aquest mòdul el recolzament aeri només contempla observacions *GPS*. Abans d'introduir observacions inercials, derives angulars i matriu de desalineament caldria implementar l'ús de quaternions per evitar la inestabilitat que genera l'ús d'angles d'Euler.

- Observacions: $<X_g, Y_g, Z_g>$ són les coordenades de l'antena *GPS* en l'instant de l'exposició.
- Paràmetres:
  - Orientació de la imatge: $<X_i, Y_i, Z_i, w_i, p_i, k_i>$
  - Offset d'antena *GPS*: $<X_a, Y_a, Z_a>$ són les coordenades a determinar d'un punt qualsevol a l'espai objecte.
- Model: ve donat per la relació entre l'observació GPS $<X_g, Y_g, Z_g>$ i l'orientació externa de la imatge $<X_i, Y_i, Z_i, w_i, p_i, k_i>$ a través d'un offset d'antena $<X_a, Y_a, Z_a>$.

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_g^{Obs} = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_i + R_i \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_a$$

4.3.2.5 Offset d'antena

L'*offset* d'antena és el vector que va des de la posició del punt principal de simetria de la càmera fins a la posició on es refereixen les observacions *GPS* (en general, el centre de fase de l'antena). En el cas que aquest grup de paràmetres s'hagi observat caldrà afegir el model per tal de considerar les observacions d'*offset*.

- Observacions: $<X_a, Y_a, Z_a>$ és el vector PPS → Antena *GPS* mesurat.
- Paràmetres: $<X_p, Y_p, Z_p>$ és el vector PPS → Antena *GPS* mesurat a determinar d'un punt qualsevol a l'espai objecte.
- Model: la relació entre paràmetres i observacions és la identitat.

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_a^{Obs} = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_a^{Par}$$

4.3.3 Models estocàstics

4.3.3.1 Desviacions a priori

Actualment *Ceres* no contempla l'ús d'una matriu de variància-covariància a priori o una matriu de pesos per ponderar les observacions. A [41] i [39] es pot veure com alguns usuaris de *Ceres* es pregunten com poden ponderar les observacions incloent-hi informació estocàstica a priori. Els desenvolupadors contesten donant alternatives a la matriu de pesos com ara l'ús de *loss functions*. Com s'ha dit anteriorment, utilitzar una *loss function* per escalar les observacions no permet aplicar desviacions diferents dins d'un mateix tipus d'observacions. És a dir, si s'aplica una *loss function* per escalar les observacions de recolzament per un factor 0.04, tots els punts de recolzament tindran aquesta desviació i no es podrà aplicar un altre valor a un punt de recolzament determinat. En cas de considerar un model estocàstic sense correlació entre observacions (matriu de variància-covariància

diagonal) la manera que proposen a [39] és escalar les matrius del sistema per l'arrel de la desviació associada a cada observació. En aquestes condicions el sistema d'equacions normals pel mètode de *Gauss-Newton*:

$$A^t PA = A^t PU$$

on *A* és la jacobiana respecte dels paràmetres i *U* és el vector de *misclosures* o cost inicial i *P* és la matriu de pesos corresponent a la matriu de variància-covariància *C*:

$$P = \begin{pmatrix} p_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & p_n \end{pmatrix} = C^{-1} = \begin{pmatrix} \frac{1}{\sigma_{11}^{\ 2}} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{1}{\sigma_{nn}^{\ 2}} \end{pmatrix}$$

es pot reescriure de la forma:

$$A'^t\ A' = A'^t\ U'$$

on per cada fila *i=1..n* de la matriu de disseny *A* i cada element del vector de *misclosures U* s'ha multiplicat per l'arrel del pes corresponent, obtenint *A'* i *U'* respectivament:

$$A' = \begin{pmatrix} \sqrt{p_{11}} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sqrt{p_{nn}} \end{pmatrix} A = \begin{pmatrix} \frac{1}{\sigma_{11}} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{1}{\sigma_{nn}} \end{pmatrix} A$$

$$U' = \begin{pmatrix} \sqrt{p_{11}} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sqrt{p_{nn}} \end{pmatrix} U = \begin{pmatrix} \frac{1}{\sigma_{11}} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{1}{\sigma_{nn}} \end{pmatrix} U$$

És a dir, cal dividir cada equació d'observació per la desviació corresponent.

En el nou mòdul basat en *Ceres*, per cada una de les observacions (fotogramètriques, recolzament, *GPS*, *offset* i pseudo-observacions dels paràmetres d'autocalibració) es considera una desviació associada per a cada component. La desviació passa a la funció de cost per tal d'escalar el residu i *Ceres* s'encarrega d'escalar la matriu dels sistema.

4.3.3.2 Desviacions a posteriori

L'anàlisi estocàstic del nou mòdul no s'ha acabat. Manca encara calcular desviacions a posteriori pels paràmetres, números de redundància, etc. Tal i com anuncien els mateixos desenvolupadors de *Ceres*, els mòduls d'anàlisi de covariàncies encara són provisionals. El maig de 2013 van publicar un mòdul anomenat *covariance.h* que permetrà continuar amb l'anàlisi estadístic del nou mòdul [38].

L'únic que s'ha dut a terme en quant a l'anàlisi d'informació estocàstica és la computació de residus per cada una de les observacions i desviació tipus global desglossant per cada grup d'observacions. A tall d'exemple es mostren els estadístics obtinguts per les observacions de punts de recolzament d'un dels blocs sintètics generats amb *AeroSint*.

```
RESIDUALS FOR GROUND CONTROL POINT OBSERVATIONS

             MINIMUM         MEAN         MAXIMUM         R.M.S.
     X    -0.0588373   -0.0030564    0.0941634    0.0382448
     Y    -0.0749399   -0.0008805    0.0501752    0.0367892
     Z    -0.0522354   -0.0010525    0.0396794    0.0258257

     Number of observation groups: 16
     Stdev.: 0.8518455
```

Per calcular els estadístics anteriors s'han aprofitat les mateixes *cost functions* que defineixen el model.

4.3.4 Estructura interna dels fitxers

El nou mòdul considera tres tipus de fitxers: paràmetres, observacions i opcions. Per tal de simplificar la lectura d'arxius es considera una entitat per línia. No hi haurà capçaleres.

4.3.4.1 Fitxers de paràmetres

Es considera un arxiu per cada tipus de grup de paràmetres. Cada línia conté un grup de paràmetres i l'identificador corresponent.

Es distingeixen dos tipus de fitxers de paràmetres, ambdós amb la mateixa estructura:

- Aproximacions inicials: fitxers d'entrada al mòdul
- Paràmetres ajustats: fitxers de sortida del mòdul

Per exemple, cada línia dels arxius *exori.pdi* i *exori.pda* contindrà:

$$IDi \quad X \quad Y \quad Z \quad \varphi \quad \kappa \quad sX \quad sY \quad sZ \quad sw \quad sp \quad sk$$

on *IDi* és l'identificador de la imatge *i* i $X\ Y\ Z$, φ, κ són els paràmetres d'orientació externa de la imatge *i*. Per últim venen les desviacions associades als sis paràmetres d'orientació externa.

4.3.4.2 Fitxers d'observacions

Es considera un arxiu per cada model. Cada línia conté un grup d'observacions, els identificadors dels paràmetres relacionats i les desviacions associades a cada observació.

Per exemple, cada línia de l'arxiu *photo.od* contindrà:

$$IDi \quad IDe \quad IDp \quad x \quad y \quad sx \quad sy$$

on *IDi*, *IDe* i *IDp* són els identificadors de la imatge *i*, el grup de paràmetres d'autocalibratge *e* i el punt objecte *p* respectivament. A continuació venen les fotocoordenades *x* i *y* i les corresponents desviacions associades *sx* i *sy*.

4.3.4.3 Fitxer d'opcions

L'arxiu d'opcions conté la parametrització de l'ajust. Cada opció s'escriu al començament d'una línia i a continuació hi ha un comentari amb la descripció de l'opció. Aquesta descripció és només un ajut a l'usuari i no es llegeix.

4.3.5 Estructures de dades

L'estructura de dades d'un programa d'ajust d'observacions cal rumiar-la amb cura ja que s'accedirà a una gran quantitat d'observacions i paràmetres, alguns dels quals de forma repetida. Si no s'optimitza el disseny de l'estructura de dades, l'eficiència i flexibilitat del programa es veuran afectades en gran mesura.

Cada grup d'observacions està relacionat amb un conjunt de grups de paràmetres. La relació grup d'observacions – grup de paràmetres és constant per cada model.

4.3.5.1 Estructura 1: orientada a entitats físiques

A continuació es defineixen els objectes que caldria crear segons una jerarquia en què les entitats físiques són els objectes pare i els grups de paràmetres i grups d'observacions són objectes descendents dels primers.

- Observació fotogramètrica
    - Grups d'observacions
    - Model i funció de cost
    - Punters a paràmetres implicats en el model: punts, orientacions i *offset*
- Punt objecte
    - Grup d'observacions: només en cas de ser punt de recolzament observat
    - Model i funció de cost
    - Grup de paràmetres
- Orientació externa
    - Grup de paràmetres
- Objecte mesura *GPS*
    - Grup d'observacions
    - Model i funció de cost
    - Punters a paràmetres implicats en el model: orientacions
- Objecte *offset* d'antena
    - Grup d'observacions
    - Model i funció de cost
    - Grup de paràmetres

Aquesta estructura de dades és poc eficient perquè l'estructura dels diferents objectes és heterogènia. Hi ha objectes com les mesures *GPS* que actuen només com a observacions i no tindran paràmetres associats. En canvi, l'objecte orientació externa només tindrà paràmetres (en general no s'observa directament). Per últim hi ha el cas d'objectes que actuen al mateix temps d'observació i de paràmetre, com ara els punts objecte i els *offsets* d'antena. Per acabar-ho d'arreglar, ni tan sols tots els punts objecte tindran observacions, només aquells que siguin de recolzament.

Una estructura de dades tan heterogènia tindrà repercussió a l'hora de muntar les matrius del sistema perquè no serà possible seguir un ordre seqüencial per accedir a les observacions. Una

alternativa seria crear un objecte auxiliar amb una relació de punters a observacions que pogués ser accedida de forma seqüencial, però aquest artifici és poc elegant i podria anar en detriment de la flexibilitat del codi.

4.3.5.2 Estructura 2: orientada a paràmetres i observacions

A continuació es defineixen els objectes que caldria crear segons una jerarquia en què els grups de paràmetres i grups d'observacions són els objectes pare i les entitats físiques són objectes descendents dels primers.

- Grups de paràmetres
  - Punts objecte
  - Orientacions externes
  - Imatges
  - *Offset*
- Grups d'observacions
  - Fotocoordenades
    - Punter a paràmetres implicats: punts, orientacions i *offset*
  - Punts de recolzament
    - Punter a paràmetres implicats: punts
  - Mesures *GPS*
    - Punter a paràmetres implicats: orientacions
  - Mesura d'*offset*
    - Punter a paràmetres implicats: *offset*
- Models i funcions de cost
  - Fotocoordenades
  - Punts de recolzament
  - Mesures *GPS*
  - Mesura d'*offset*

Aquesta estructura és molt més homogènia. A més, les observacions es poden accedir seqüencialment de forma natural, només cal repassar els grups d'observacions un a un i, per cada un, accedir directament als paràmetres implicats.

4.3.6 Les classes del nou mòdul

Un cop escollida l'estructura de dades cal dissenyar l'estructura de classes que la materialitzarà. Aquí entraran en joc els coneixements de programació orientat a objectes a l'hora de triar una estructura o una altra. Donada l'estructura de dades orientada a paràmetres i observacions, es proposen les dues materialitzacions següents:

1. Estructura plana: una classe observació, paràmetre i funció de cost per cada entitat
2. Estructura jeràrquica: fent ús de l'herència de classes (no implementat)

4.3.6.1 Estructura de classes 1: estructura plana

- Grups de paràmetres: *my_parameter_blocks.h*

  L'estructura de la classe de grups de paràmetres és clara. Els membres són identificador i grup de paràmetres.

```
class CPuntPar{
  int ID;
  double X, Y, Z;
};

class CFotoPar{
  int ID;
  double X, Y, Z, w, p, k;
};

class CEbnerPar{
  int ID;
  double df, dx0, dy0, e[12];
};

class COffsetPar{
  int ID;
  double X, Y, Z;
};
```

- Grups d'observacions: *my_observation_blocks.h*

  L'estructura de la classe grups d'observacions conté els valors de les observacions, desviacions associades i residus. A més, per identificar cada grup d'observacions, es defineix un conjunt de punters a les classes de grups de paràmetres implicats, un conjunt d'observacions i un conjunt de desviacions associades. L'ús d'un identificador propi per aquests objectes és redundant perquè els punters a tots els paràmetres implicats són suficients per identificar inequívocament cada observació.

```
class CFotogramObs{
  CFotoPar* IDf, CPuntPar* IDp, CEbnerPar* IDe;
  double x, y;
  double sx, sy;
  double rx, ry;
};

class CPuntSuportObs{
  CPuntPar* IDp;
  double X, Y, Z;
  double sX, sY, sZ;
  double rX, rY, rZ;
};

class CEbnerObs{
  CEbnerPar* IDe;
```

```
  double df, x0, y0, e[12];
  double sdf, sx0, sy0, s[12];
};

class CGPSObs{
  CFotoPar* IDf, COffsetPar* IDa;
  double X, Y, Z;
  double sX, sY, sZ;
  double rX, rY, rZ;
};

class COffsetObs{
  COffsetPar* IDa;
  double X, Y, Z;
  double sX, sY, sZ;
  double rX, rY, rZ;
};
```

- Models i funcions de cost: *my_cost_functions.h*

  La classe de models i funcions de cost només conté dos tipus de membres: els punters a les classes de grups d'observacions i a les classes de grups de paràmetres. *T cf(...)* és un mètode que permet calcular el cost a partir d'observacions i paràmetres i retorna una variable de tipus *T (template)* [11].

```
class CFotogramObsCF{
  CFotogramObs* pObsGroup;
  CFotoPar* IDf, CPuntPar* IDp, CEbnerPar* IDe;
  T cf(CFotogramObs*, CFotoPar*, CPuntPar*, CEbnerPar*);
};

class CPuntSuportObsCF{
  CPuntSuportObs* pObsGroup;
  CPuntPar* IDp;
  T cf(CPuntSuportObs*, CPuntPar*);
};

class CEbnerObsCF{
  CEbnerObs* pObsGroup;
  CEbnerPar* IDe;
  T cf(CEbnerObs*, CEbnerPar*);
};

class CGPSObsCF{
  CGPSObs* pObsGroup;
  CFotoPar* IDf, COffsetPar* IDa;
  T cf(CGPSObs*, CGPSObs*, COffsetPar*);
};
```

[11] El tipus de retorn *T*emplate per les funcions de cost s'explicarà més endavant a l'apartat *4.3.6.3 Templates. Per què ens calen?*

```
class COffsetObsCF{
  COffsetObs* pObsGroup;
  COffsetPar* IDa;
  T cf(COffsetObs*, COffsetPar*);
};
```

Els objectes de la classe *cost function* materialitzen el model matemàtic (funcional i part de l'estocàstic).

- Classe bloc d'aerotriangulació: *my_at_block.h*

  Per últim s'ha definit una classe que farà d'interfície entre el mòdul principal i les classes de grups d'observacions i de grups de paràmetres. Al mòdul principal es crearà un objecte de la classe *CBlocAT* que contindrà punters a objectes de tipus *parameter block* i *observacion block* i també contindrà mètodes per carregar i imprimir dades en arxius i interactuar amb llibreries *Ceres*.

  A continuació es mostra una simplificació de la definició de la classe, obviant les variables membre i mètodes poc rellevants per l'explicació mitjançant punts suspensius.

```
class CBlocAT
{
  // Mètodes de càrrega de paràmetres i observacions
  int load_punts_par(const char* file_name);
  int load_exori_par(const char* file_name);
  int load_ebner_par(const char* file_name);
  int load_offset_par(const char* file_name);
  int load_control_obs(const char* file_name);
  int load_photocoord_obs(const char* file_name);
  int load_ebner_obs(const char* file_name);
  int load_gps_obs(const char* file_name);
  int load_offset_obs(const char* file_name);

  // Mètode d'impressió de paràmetres ajustats a arxiu
  bool print_punts_par(const char* file_name);
  bool print_exori_par(const char* file_name);
  bool print_ebner_par(const char* file_name);
  bool print_offset_par(const char* file_name);

  // Mètode d'impressió de residus a arxiu
  bool print_photocoord_res(const char* file_name);
  bool print_control_res(const char* file_name);
  bool print_gps_res(const char* file_name);
  bool print_offset_res(const char* file_name);

  // Mètodes per crear objectes cost function
  bool add_photocoord_CF(Ceres::problem* problem);
  bool add_control_CF(Ceres::problem* problem);
  bool add_ebner_CF(Ceres::problem* problem);
  bool add_gps_CF(Ceres::problem* problem);
  bool add_offset_CF(Ceres::problem* problem);
```

```
    // Mètode per cridar la rutina d'ajust de Ceres
    bool run_adjustment();

    // Membres: punters a objectes de grups de paràmetres
  i observacions
    CPuntPar* punt_par;
    CFotoPar* foto_par;
    CEbnerPar* ebner_par;
    COffsetPar* offset_par;
    CPuntSuportObs* control_obs;
    CFotogramObs* foto_obs;
    CEbnerObs* ebner_obs;
    CGPSObs* gps_obs;
    COffsetObs* offset_obs;

  }
```

Aquesta és l'estructura de classes implementada al nou mòdul *Ceres*. S'ha escollit perquè l'accés a les observacions és seqüencial i l'accés als paràmetres relacionats amb cada grup d'observacions és directa gràcies als punters que guarda cada grup d'observacions. És a dir, no cal recórrer tots els paràmetres cada cop que es vol accedir des d'un grup d'observacions.

4.3.6.2 Estructura de classes 2: estructura jeràrquica

Si s'analitza l'estructura anterior es pot veure clarament que hi ha patrons repetitius en la definició de la classe *CBlocAT*. Aquest patrons repetitius es tradueixen en una duplicació de codi a l'hora de fer tasques comunes com ara la impressió de resultats de dos grups de paràmetres diferents.

Per exemple, els quatre mètodes de càrrega des d'arxiu de *CBlockAT*

```
    int load_punts_par(const char* file_name)
    int load_exori_par(const char* file_name)
    int load_ebner_par(const char* file_name)
    int load_offset_par(const char* file_name)
```

s'haurien d'implementar amb un únic mètode genèric que s'adapti al nombre de paràmetres de l'objecte grup de paràmetre que el cridi.

```
    int load_par(const char* file_name)
```

En totes les classes de grups de paràmetres existeix un membre que és l'identificador i tot seguit un conjunt d'*n* paràmetres. Donat que totes les observacions i paràmetres es poden considerar d'un mateix tipus, llavors és possible agrupar-los en vectors de dimensió *n*. De la mateixa manera, en el cas de les classes de grups d'observacions hi haurà *m* observacions i *m* desviacions associades que es poden agrupar en vectors ambdós de dimensió *m*. Finalment caldrà considerar un vector que contingui *n* punters (un per cada grup de paràmetres relacionats).

A l'hora de definir els grups de paràmetres de punts caldrà definir les següents classes:

```
  class CParameterGroup {
     CParameterGroup(int iNumParams){
```

```
            m_iNumParams = iNumParams;
            m_pdParams = new double[iNumParams];
        }
        ... // Definició de membres, etc.
    }

    class CPointPG : public CParameterGroup {
        CPointPG() : CParameterGroup(3) {};
    }

    class CParameterSet {
        void LoadFromFile(string nom_arxiu)
        CParameterGroup* m_ParameterGroups;
    }

    class CPointPS : public CParameterSet{
        virtual CParameterGroup* GetNewParameterGroup() {
            return new CPointPG();
        }
    }
```

A partir d'elles es crearà un únic objecte *CPointPS* que hereta de *CParameterSet* i que contindrà els membres de tipus *CPointPG* (que al seu torn hereta de *CParameterGroup*).

```
    CPointPS PuntsObjecte;
    PuntsObjecte.LoadFromFile("input.txt");
```

A les figures 4.3 i 4.4 es presenta el diagrama per una proposta del prototipus complert. Cada caixa és una classe que conté membres en blau i mètodes en vermell. Les fletxes vermelles mostren les relacions d'herència i van des de les classes "filles" fins a les classes "pare". Les fletxes verdes van des d'una variable punter fins a la variable a la qual apunten.

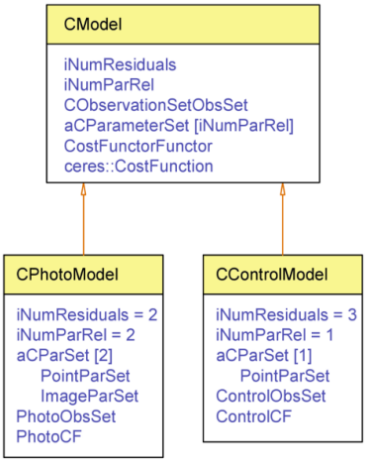


Figura 4.3

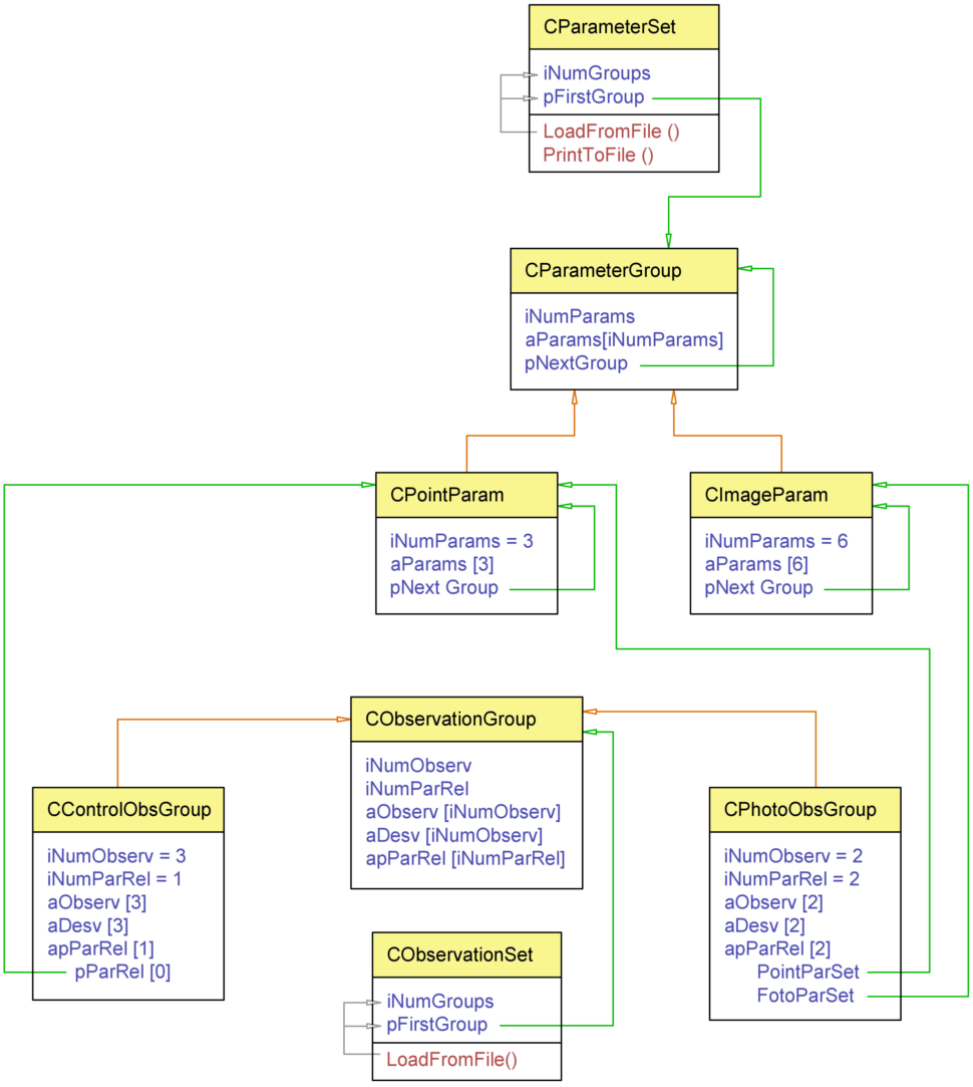


Figura 4.4

L'estructura jeràrquica es proposa amb la intenció de solucionar els patrons repetitius de l'estructura plana. El resultat seria aproximadament igual d'eficient que l'estructura plana, però guanyant en flexibilitat a l'hora d'implementar nous models. No s'ha implementat al nou mòdul basat en *Ceres*, però s'aconsella dur-la a terme abans d'ampliar el mòdul amb nous models.

4.3.6.3 Templates. Per què ens calen?

Els *templates* són un tipus de dades en programació que permeten implementar una única funció o classe que s'adapti automàticament (sense duplicar codi) a diferents tipus de dades. És a dir, ha de permetre crear algorismes que siguin independents del tipus de dades que utilitza. Per exemple, per fer que la funció quadrat(x) accepti valors enters, reals i reals de doble precisió un mètode seria sobrecarregar innecessàriament la funció:

```
int quadrat(int x) { return x*x; }
float quadrat(float x) { return x*x; }
double quadrat(double x) { return x*x; }
```

Per estalviar aquesta duplicació de codi, es pot reescriure quadrat(x) definint tant la variable d'entrada com la sortida, com tipus *template*:

```
template <class T>
T quadrat(T x) { return x*x; };
```

La diferenciació automàtica amb *Ceres Solver* fa ús de *templates* per definir un tipus de dades especials, els nombres duals (veure l'apartat *2.3.5. Tècniques de diferenciació*). És per aquest motiu que caldrà familiaritzar-se amb l'ús de *templates* en el nou mòdul amb l'objectiu d'aprofitar el potencial de la diferenciació automàtica.

4.3.7 Mètodes robustos

De cara a una continuació d'aquest treball es proposa implementar les *loss functions* següents, ja que són els dos tipus de grups d'observacions que acostumen a contenir errors grollers en fotogrametria. Per cada una caldrà escollir un dels mètodes robustos implementats a *Ceres* o bé un de definit per l'usuari.

- *Photo LossFunction*: detecció d'errors grollers en les fotocoordenades. Són els errors grollers més comuns i es deriven de problemes amb la correlació automàtica (ombres, objectes en moviment, aigua, etc.) o problemes deguts mesures manuals (equivocacions humanes en la identificació de punts homòlegs).
- *Control LossFunction*: detecció d'errors grollers en les observacions de punts de recolzament. Una altra font d'errors grollers menys comú són les observacions a camp de punts de recolzament. Per exemple, un punt de recolzament mesurat en la cantonada equivocada d'una teulada, un topògraf que s'hagués descuidat d'aplicar l'ondulació del geoide, ambigüitats mal resoltes en l'ajust de bases *GPS*, etc.

Implementar mètodes robustos a *Ceres* és qüestió de pocs segons. Per exemple, al tutorial de *Ceres* que es troba a [2] s'explica que per implementar el mètode robust *CauchyLoss* l'únic que cal fer és canviar:

```
problem.AddResidualBlock(cost_function, Null, &par1, &par2);
```

per:

```
problem.AddResidualBlock(cost_function, Null, &par1, &par2);
```

Gran part de la dificultat rau en el fet de saber quin mètode particular és el que més bé s'adapta al problema que es vol resoldre i, un cop triat, parametritzar-lo adequadament tampoc és trivial.

En aquest treball s'ha escollit implementar la *Huber loss function* únicament perquè és el mètode que *Google* ha triat pel seu mòdul *bundle_adjuster.cc*. Més endavant caldrà escollir el mètode més adient pel nou mòdul amb més criteri.

Les proves d'avaluació de *Ceres* en aquest treball posen de manifest que és crucial el fet de saber aplicar-los correctament. Per exemple, suposem que existeix un error groller en l'observació d'un punt de recolzament. Si s'apliquen mètodes robustos sense un criteri assenyat, i el punt de recolzament està observat en poques imatges, la gran diferència d'escala entre observacions fotogramètriques i de recolzament farà que es descarti l'observació fotogramètrica i el punt de recolzament erroni quedi amb un residu proper a zero.

5 Part IV. Avaluació del nou mòdul

Existeixen dues maneres d'avaluar l'exactitud del nou mòdul:

- Comparar els resultats contra un software ja validat
- Validar directament per mitjà de dades sintètiques

5.1 *ACX*. Un software validat

La primera referència és el software vigent a l'*ICGC*, l'*ACX*. Aquest software ha servit per ajustar xarxes geodèsiques i fotogramètriques a l'*ICGC* durant 20 anys tant en projectes de desenvolupament com de producció. En el cas fotogramètric, l'ús continuat amb blocs diversos ha permès depurar el codi, posar-lo a prova i fer millores. El resultat és un software molt estable i àmpliament "testejat" que servirà per fer una primera validació del nou mòdul.

L'avantatge principal d'*ACX* de cara a l'avaluació del nou mòdul és que permet extreure informació detallada de cada iteració. En particular, es poden imprimir les matrius del sistema linealitzat (la jacobiana, la diagonal de la matriu de pesos i el vector de termes independents) i també les matrius del sistema normalitzat (matriu normal i vector de termes independents normalitzat).

5.2 *AeroSint*. El generador de dades sintètiques

Malgrat *ACX* té una extensa validació, la manera més rigorosa d'avaluar un software d'ajust d'observacions és a través de l'ús de dades sintètiques. *AeroSint* és una eina de auxiliar per generar dades sintètiques que s'ha desenvolupat de cara a obtenir blocs de test que serviran únicament per avaluar el nou mòdul.

El principi és molt simple d'enunciar: *donat un problema d'ajust d'observacions, les observacions sintètiques són aquelles que s'han calculat a partir dels paràmetres segons un model matemàtic considerat*. Per materialitzar-ho cal dur a terme els següents passos:

1. Escollir la configuració del vol, del terreny i de la càmera
2. Calcular paràmetres en base a la configuració de vol
3. Calcular observacions a partir dels paràmetres
4. Afegir soroll gaussià
5. Exportació d'observacions i paràmetres

5.2.1 Configuració del vol, terreny i càmera

S'han escollit els paràmetres de configuració més representatius i còmodes a l'hora de dissenyar un nou bloc. Per exemple, en comptes de que l'usuari hagi d'entrar l'alçada de vol és millor que entri la mida de píxel sobre el terreny perquè és un paràmetre més representatiu. De la mateixa manera, s'ha escollit que l'usuari triï la distribució matricial de punts de suport en comptes de les coordenades de tots ells per comoditat. La millor manera d'il·lustrar la necessitat de que els paràmetres siguin "còmodes" és a través d'un exemple. Només cal imaginar el fet d'haver dissenyat un bloc de 500 imatges i inventar-se les coordenades dels 25 punts de suport. Inventar-se 25x3 = 75 valors per les coordenades que estiguin homogèniament distribuïts pel bloc? Cap problema. Però si just després es vol fer un altre test en què, en comptes de 500 imatges siguin 3000 i haver de re-inventar les coordenades dels punts de suport (que ara poden ser prop de 100x3 = 300), ja deixa de ser tan trivial.

La configuració del vol es llegeix d'un arxiu com el següent:

```
3                    # Nombre de passades
10                   # Nombre d'imatges per passada
60 30                # Solapament longitudinal i transversal
0.075                # Mida de píxel terreny (m)
400000 4630000       # Coord. 1a imatge , 1a passada
3 3                  # Files i columnes de punts sintètics
3 3                  # Distribució de GCPs
0.0 0.0 70.0         # Roll, pitch i heading promig (gons)
1.20 -0.50 0.80      # Offset d'antena (m)
2.00 2.00            # Desviació fotocoordenades (micres)
0.04 0.04 0.04       # Desviació obs recolzament (m)
0.05 0.05 0.05       # Desviació obs GPS (m)
0.03 0.03 0.03       # Desviació obs offset d'antena (m)
```

De la mateixa manera, la configuració de la càmera:

```
0.120                # Distància focal nominal (m)
12.0                 # Mida de píxel en espai imatge (micres)
13824 7680           # Format d'imatge: num_files num_cols
0.0 0.0              # Punt principal de simetria
1                    # Correcció de focal: ON(1) OFF(0)
1                    # Correcció del PPS: ON(1) OFF(0)
1                    # Ús de paràmetres d'Ebner: ON(1) OFF(0)
0.000009             # Correcció de distància focal (m)
0.000005 0.000008    # Correcció del punt ppal de simetria (m)
0.00006 0.000009 -0.0003 0.0007 -0.0002 0.0003  # Ebner 1-6
0.009 -0.004 0.006 0.005 -0.200 0.300           # Ebner 7-12
```

Per últim, la configuració del terreny es llegeix d'un arxiu similar. No es mostra. Només cal saber que hi consta una alçada promig del terreny *Hm* i els coeficients *a*,*b* de la funció d'una superfície del tipus $H = f(x, y)$.

5.2.2 Paràmetres calculats

A partir de la configuració de vol, càmera i terreny s'han calculat directament els centres de projecció, l'offset d'antena i els paràmetres d'autocalibratge d'*Ebner*. Els centres de projecció es distribueixen en un bloc regular de *ndp* passades i *nipp* imatges per passada. Les rotacions *w, p, k* es calculen a partir dels valors de *roll, pitch, heading* promig de la configuració del vol, afegint soroll gaussià i fent els canvis d'eixos i sentits corresponents i considerant una matriu de desalineament identitat.

Les coordenades terreny dels punts de lligam es poden obtenir de dues maneres:

1. Sobre el terreny: cal determinar primer els límits de la regió que ocupa el bloc fotogramètric sobre el terreny (*footprint*) i després fer una quadrícula de punts amb l'interval desitjat. El problema d'aquest mètode és que el càlcul del *footprint* es pot complicar ràpidament si es consideren blocs irregulars o amb imatges de perspectives que no siguin estrictament *nadirals*.

2. Projectant des de les imatges: l'alternativa és fer una distribució matricial de *m* files i *n* columnes de fotocoordenades sobre les imatges per després projectar per resecció directa sobre el terreny. És l'opció escollida per *AeroSint*.

5.2.3 Observacions calculades

Es tracta de calcular observacions fent servir el mateix model funcional que utilitza el nou mòdul a l'hora de resoldre l'ajust d'observacions. La generació de paràmetres i observacions es fa conjuntament aprofitant els mateixos bucles. El procediment seguirà el següent ordre:

1. Per cada grup de paràmetres d'*offset* es generarà el corresponent grup d'observacions també considerant un model funcional identitat (observació = paràmetre).
2. Per cada imatge es crea el grup de paràmetres d'orientació externa i alhora es crea el corresponent grup d'observacions *GPS*. Caldrà considerar els paràmetres d'*offset* que relaciona orientació externa amb observació *GPS*.
3. Tal i com s'ha dit a l'apartat *5.2.2 Paràmetres calculats,* els punts objecte es projectaran des de les imatges seguin una distribució matricial de *m* files i *n* columnes.
4. Per cada localització on s'hagi dissenyat un punt de recolzament es crearà un nou punt objecte i es marcarà com a punt de recolzament. Al mateix moment es crearà el corresponent grup d'observacions. Com que el model funcional és la identitat no caldrà fer intervenir cap més paràmetre.
5. Per cada grup de paràmetres d'autocalibratge es crearan les pseudo-observacions corresponents. En general es deixen les observacions a zero.
6. Per cada punt objecte es calcularan per resecció inversa les fotocoordenades sobre les imatges. Aquí caldrà tenir en compte els grups de paràmetres de punts, orientacions i autocalibratge.

La figura 5.1 simbolitza una xarxa fotogramètrica de 4 passades horitzontals generada amb *AeroSint*. En blau es mostren les connexions entre punts i orientacions. Els 9 punts vermells representen els punts de recolzament.

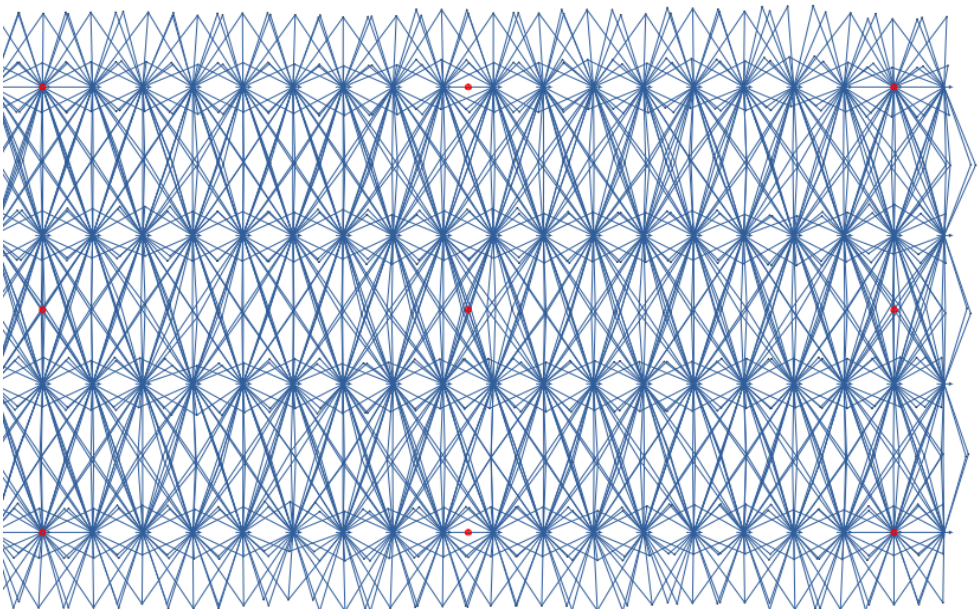


Figura 5.1

5.2.3.1 Neteja d'observacions

Les mateixes observacions fotogramètriques serviran també per determinar si els punts entren o no en les imatges. És a dir, primer es calculen observacions per totes els punts en totes les imatges possibles i després es descarten aquells punts que tinguin fotocoordenades fora dels llindars definits pel format d'imatge considerat. Aquest filtrat cal dur-lo a terme després de la generació de soroll gaussià i de l'aplicació de correccions de distorsió, autocalibratge, etc. S'ha escollit aquest ordre per no aplicar soroll a observacions que acabaran sent descartades ja que la generació de soroll gaussià és el procés de major cost de computació a *AeroSint*.

Un cop generada la xarxa fotogramètrica es farà un segon filtrat que consisteix en eliminar punts que apareixen en menys de dues imatges i punts de recolzament que tinguin coordenades, però que no arribin a sortir en cap imatge. Aquests casos són poc freqüents, però cal considerar-los ja que el soroll gaussià i autocalibratge s'apliquen a les fotocoordenades després d'haver creat els paràmetres corresponents. S'evita així l'existència de grups de paràmetres aïllats o que no siguin determinables a través de la xarxa.

5.2.3.2 Un greu problema de sintetització

Un cop es va donar per tancat *AeroSint* i després d'estar més que convençut de que les observacions generades eren 100% sintètiques i amb residus de distribució gaussiana perfectament controlats es van iniciar els testos de validació del nou mòdul. Els primers testos amb blocs petits mostraven resultats perfectes i amb *RMS* proper a la desviació associada a cada grup d'observacions amb qualsevol configuració.

La sorpresa va sorgir a l'hora de generar blocs sintètics de grans dimensions per posar a prova els límits del nou mòdul. En blocs de més de 1000 imatges on es considera l'ús de paràmetres d'autocalibratge, tant *ACX* com *Ceres* van passar a no convergir malgrat s'escollís una aproximació inicial igual al valor ajustat. Analitzant els fitxers d'entrada es van detectar observacions fotogramètriques que entraven en imatges de l'altre extrem del bloc. El problema era clarament *AeroSint*. L'origen del problema era que, a l'hora de filtrar quines observacions fotogramètriques apareixen en la imatge i quines queden fora, els mateixos paràmetres d'autocalibratge aplicaven distorsions "enormes" pels punts que tenen fotocoordenades virtuals molt lluny del centre de projecció. Aquests punts no haurien de sortir mai en la imatge, però les mateixes correccions d'autocalibratge feien que sí entressin.

La solució és fàcil d'implementar. La correcció per autocalibratge cal aplicar-la després d'haver comprovat si el punt entra o no en la imatge. La neteja d'observacions ja s'encarregarà d'eliminar l'observació si l'autocalibratge és la causa de que deixi d'entrar en la imatge.

5.2.4 Soroll gaussià

La generació de soroll gaussià es basa en el teorema central del límit. La funció *Gauss(rms)* retorna una variable pseudoaleatòria de distribució normal centrada en el zero, desviació tipus $\sigma = 1.0$ i error mitjà quadràtic *rms* a partir d'una variable també pseudoaleatòria, però de distribució equiprobable retornada per la funció *rand()*.

Per tal de controlar els decimals retornats per la funció *Gauss(rms)* s'ha sobrecarregat la funció per que retorni un nombre amb una resolució *res* a escollir.

```
double Gauss(double rms, int res)
{
  int m = pow(10, res);
  return round(Gauss(rms*m))/m;
}
```

5.2.5 Exportació d'observacions i paràmetres

Un cop generades les observacions i paràmetres del bloc sintètic, s'exporten a dos formats de fitxers. Els arxius de sortida d'*AeroSint* seran els arxius d'entrada dels programaris d'ajust a avaluar, i.e. el nou mòdul *Ceres* i *ACX*. L'exportació de les mateixes dades sintètiques a dos programaris diferents permetrà que siguin avaluats sota les mateixes condicions.

5.3 *Compara*. El comparador de mòduls

L'avaluació del nou mòdul requerirà fer multitud de comparacions entre els resultats obtinguts per dos mòduls diferents i també entre dues iteracions consecutives del mateix mòdul. Per aquest treball s'ha desenvolupat una senzilla eina anomenada *Compara* que permet estalviar-se haver de fer comparacions manuals entre arxius que a la llarga resultarien feixugues i difícilment serien exhaustives. L'eina *Compara* calcula distàncies entre dues solucions en l'espai de paràmetres. En concret, calcula diferències en les coordenades terreny per un mateix punt en ajustos diferents.

Els casos avaluats són els següents:

- Considerant només un mòdul:
    - Diferència entre iteracions successives: permet calcular de forma indirecta el *pas* calculat, és a dir, el vector de correccions obtingut com a solució del sistema lineal corresponent a una iteració concreta.
    - Diferència entre paràmetres ajustats i aproximacions inicials: permet calcular la distància "en línia recta"[12] recorreguda per l'espai de paràmetres al llarg de tot el procés iteratiu.
    - Diferència entre els paràmetres ajustats i la referència: es calcula l'exactitud de les coordenades terreny ajustades prenent com a veritable valor aquell que s'extreu de l'eina *AeroSint*. En bloc sintètics sense soroll gaussià no s'espera trobar diferències.
- Considerant dos mòduls:
    - Diferència entre els paràmetres ajustats: un cop s'ha assolit cert criteri de convergència en ambdós mòduls. Evidentment, si algun dels dos no convergeix no es compararan els mòduls.

Per cada fitxer d'entrada es detecta quin format de dades té cada un, *Ceres* o *ACX*. Acte seguit es fa una anàlisi sintàctica en base al format detectat i s'emmagatzemen els paràmetres en dos objectes de tipus vector de *doubles*, un per cada conjunt de dades. Finalment es crea un tercer vector de *doubles* que contindrà la resta dels dos vectors anteriors. Per restar vectors es recorre arbitràriament un dels vectors i, per cada punt, es busca l'identificador corresponent al segon vector. En cas de trobar-se s'afegeix la diferència al tercer vector. La sortida del programa consisteix en els següents fitxers:

- Llistat de diferències: un fitxer de text amb una línia per cada punt objecte comú als dos fitxers d'entrada. Conté l'identificador i les diferències desglossades per components.
- Resum d'estadístics: un fitxer amb alguns dels estadístics que permetran valorar la magnitud de les diferències entre solucions. A continuació es mostra un exemple del

[12] No se sumen les distàncies parcials corresponents a cada iteració. Únicament s'obté la resta entre dos vectors: aproximació inicial i solució final.

contingut del fitxer d'estadístics on s'analitza l'exactitud de la solució obtinguda amb *Ceres*. L'arxiu *punts.chk* és la referència obtinguda per *AeroSint* i l'arxiu *punts-6.pda* són els paràmetres dels punts objecte ajustats obtinguts a la 6a iteració de *Ceres*.

```
Input files to be compared:
  1st file name: punts.chk
  2nd file name: punts-6.pda

Comparing object point parameters:
  Points in 1st file:   1404
  Points in 2nd file:   1420
  Points in common:     1404

           MINIMUM            MEAN          MAXIMUM          R.M.S.
  X    -0.0842630000   -0.0168089701    0.0436900000    0.0224785529
  Y    -0.1061209999    0.0068538953    0.1030019997    0.0215117900
  Z    -0.1561650000    0.0194029516    0.2309280000    0.0503573040

Maximum difference between two object point parameters: 0.2309280000
Overall RMS for object point parameters:       0.0341755298
```

La diferència que s'observa entre el nombre de punts d'un i altre fitxer és deguda a que només es comparen els punts de xec. Els 16 punts que hi ha de més a *punts-6.pda* són els punts de recolzament del bloc sintètic.

5.4 *Ceres*. Avaluant el nou mòdul

Es posarà a prova el nou mòdul en base a un conjunt de requisits que són desitjables en qualsevol programari d'ajust de feixos. Idealment el nou mòdul ha de ser estable, exacte, amb control estocàstic rigorós, robust, eficient, genèric i flexible. A [8] es proposa una llista molt més complerta que inclou altres requisits com la simplicitat i el fet d'evitar els set pecats de desenvolupament de software: rigidesa, fragilitat, immobilitat, viscositat, complicació innecessària, repetició innecessària i opacitat. Aquest treball però es limitarà a l'estudi dels set primers.

5.4.1 Estabilitat

L'estabilitat és la capacitat de convergir des d'un rang ampli d'aproximacions inicials. Les estratègies basades en mètodes de regió de confiança com ara *LM* i *DL* són conegudes per la seva estabilitat a diferència d'altres més ràpides, però menys estables com *GN*. Precisament, *Ceres* permet escollir entre les dues primeres estratègies (*LM* i *DL*) i, en canvi, *ACX* només considera l'estratègia de *GN*. Malgrat això, la primera prova efectuada a l'apartat *5.5 Jugant amb Ceres* posa de manifest que *Ceres* és poc estable amb desviacions properes a zero. En aquest sentit, *Ceres* necessita efectuar moltes iteracions mentre *ACX* assoleix la convergència en tan sols una iteració.

De cara a avaluar l'estabilitat, s'ha generat un bloc sintètic i posteriorment s'han modificat les aproximacions inicials allunyant-les de la solució.

El bloc sintètic consta de 400 imatges amb solapament longitudinal del 60% i transversal del 40%. Les orientacions externes es belluguen uns 2º de la posició nadiral utilitzant soroll gaussià. S'aplica un offset d'antena determinat. Les imatges simulades imiten la geometria d'una càmera digital de 120mm de distància focal amb una mida de píxel sobre el terreny resultant de 7.5cm. S'inclouen distorsions a les imatges que s'absorbiran automàticament per mitjà d'un grup

d'autocalibratge d'en *Ebner*. L'ajust resultant consta de 24151 grups d'observacions, 5972 punts de canemàs i 16 punts de recolzament observats.

- Simulació T01: s'ha aplicat un desplaçament de 10km cap a l'Est a les aproximacions inicials de les orientacions externes de la passada 1. Un desplaçament de 10km no és arbitrari. S'ha escollit aquest valor en concret perquè, després d'un seguit de proves amb diferents valors, és aquí on s'han pogut apreciar diferències d'estabilitat entre programaris.

  ACX peta per culpa d'aquest desplaçament inicial. La sortida del programa permet veure que aconsegueix fer una iteració i durant la resolució del sistema lineal de la segona iteració peta.

  Pel que fa a Ceres s'ha provat amb dues estratègies de resolució diferents:

  - *Dog-Leg*: amb aquesta estratègia Ceres no arriba a petar, però tampoc convergeix en un termini raonable. Al llarg del procés iteratiu es pot veure com a les primeres iteracions va alternant diferents tipus de *pas* (*Cauchy*, *Newton* i *Dog-Leg*) sempre dins de l'estratègia *Dog-Leg.* En cert punt, tant el *pas* com el radi de la regió de confiança s'estabilitzen i entra en un bucle aparentment infinit amb un valor de correccions que també es manté quasi constant. Si s'analitza amb cura es veu com, al llarg de 3000 iteracions la funció de cost retorna valors cada cop més petits, però amb una progressió aparentment asimptòtica i molt lluny de la solució.
  - *Levenberg-Marquardt*: a diferència dels resultats que obtén Lourakis et al. a [18], aquí s'han obtingut millors resultats amb l'estratègia de *Levenberg-Marquardt*. En tan sols 9 iteracions *Ceres* assoleix un mínim i s'hi estabilitza amb correccions inferiors a 10-6, gradient de 0.00 i un *RMS* pels residus de l'ordre de la desviació de les observacions corresponents. De fet la solució a la que arriba és equivalent a la que s'obté partint d'aproximacions inicials sense pertorbar.

  En general, les conseqüències d'aplicar més desplaçament encara que sigui només en una passada són fatals tant per *ACX* com per *Ceres*. També el fet de modificar la resta d'aproximacions inicials farà que fàcilment deixin de convergir els dos programaris. Caldria continuar fent simulacions més ben distribuïdes per tota la xarxa i de caire menys sistemàtic.

5.4.2 Exactitud

L'exactitud és la discrepància entre els paràmetres ajustats i el valor vertader dels mateixos. En ajustos a partir de dades reals, serà impossible conèixer el valor vertader, però de cara a avaluar el nou mòdul tenim dues alternatives, una suposar que el valor vertader dels paràmetres és aquell obtingut per un software validat i l'altra crear un ajust amb dades sintètiques.

A l'hora de comparar el resultat obtingut al final del procés iteratiu no és necessari establir un criteri de convergència comú als dos softwares. L'important és que s'hagi estabilitzat al menys fins a cert llindar, independentment de si s'han fet les iteracions estrictament necessàries o 200 de més.

S'han dut a terme dues simulacions, una amb model estocàstic i una sense. El bloc sintètic consta de 400 imatges i té una configuració idèntica a la de l'aparat *5.4.1 Estabilitat*.

- Simulació T02a: es consideren observacions amb desviació zero per tal de veure si els softwares convergeixen a la solució exacta.

Com era d'esperar, tant per l'execució amb *ACX* com per *Ceres* els errors màxims són de 10^{-6} i amb un RMS de $4 \cdot 10^{-8}$. Les diferències són degudes a errors d'arrodoniment.

Comparant *Ceres* amb *ACX* s'obtenen diferències nul·les per tots els punts (inferiors a la resolució de 10^{-10}).

- Simulació T02b: un cop garantit que, en absència de soroll, el resultat és exacte es repeteix l'experiment afegint soroll a les observacions. A les fotocoordenades s'aplica una desviació de 2μm, als punts de recolzament 4cm, a les mesures GPS 5cm i a la mesura d'offset d'antena 3cm.

  El soroll aplicat a les observacions fa que no es puguin esperar diferències nul·les respecte de la referència creada per *AeroSint*. El que sí s'espera és que tinguin un RMS similar a la desviació associada pel recolzament, però incrementat en cota degut a la pitjor determinabilitat (per la gran correlació entre la cota i la paral·laxi).

  Efectivament,

	MINIMUM	MEAN	MAXIMUM	R.M.S.
X	-0.0430530000	0.0092287867	0.0675560000	0.0141102188
Y	-0.0933200000	-0.0036132705	0.0884260004	0.0126497198
Z	-0.2160500000	-0.0159102085	0.1752010000	0.0364299012

  Les diferències entre Ceres i ACX continuen sent estrictament zero.

5.4.3 Control estocàstic

Tal i com s'explica a l'apartat *4.3.3 Models estocàstics,* el nou mòdul encara no té implementat el càlcul de desviacions estàndard a posteriori pels paràmetres, números de redundància i altres estadístics imprescindibles per a un bon control dels paràmetres ajustats. Aquesta tasca es durà a terme gràcies a una ampliació que *Google* ha fet recentment a la versió oficial de *Ceres Solver*, la classe *covariance* [38]. A data d'avui, la informació estocàstica que s'extreu a l'arxiu d'estadístics *.sta* (residus màxims, mínims, promig i *RMS*) no és suficient per un control estocàstic rigorós.

5.4.4 Robustesa

La robustesa és la capacitat de detectar errors grollers i eliminar-los de l'ajust per tal de que no afectin a la solució.

A continuació es faran quatre simulacions a partir d'un mateix bloc. Introduirem un error groller a les observacions d'un dels punts de recolzament i analitzarem quin mètode permet detectar l'error groller alhora que minimitzi l'impacte sobre la resta d'observacions.

El bloc sintètic consta de 520 imatges amb solapament longitudinal del 60% i transversal del 60%. Les orientacions externes es belluguen uns 2º de la posició nadiral utilitzant soroll gaussià. S'aplica un offset d'antena determinat. Les imatges simulades imiten la geometria d'una càmera digital de 120mm de distància focal amb una mida de píxel sobre el terreny resultant de 7.5cm. S'inclouen distorsions a les imatges que s'absorbiran automàticament per mitjà d'un grup d'autocalibratge d'en *Ebner*. L'ajust resultant consta de 41806 grups d'observacions, 7793 punts de canemàs i 20 punts de recolzament observats.

- Simulació T03a:

  L'error groller consistirà en sumar dos metres a l'observació de la cota del punt de recolzament 304 i alhora sumar 20μm a la component *x* de la fotocoordenada del punt

de canemàs 7039008 observat a la imatge 8038. S'han escollit aquests dos punts perquè en ser molt propers entre ells (uns 25m en l'espai objecte) un error groller en un pot afectar fàcilment a l'altre i a l'inrevés.

- Buscant l'error groller amb *ACX*.

  Si s'analitza l'RMS per components dels residus dels punts de recolzament es detecta un primer indici. L'RMS és similar a la desviació del recolzament en planimetria, però en cota és 10 vegades superior.

  RMS X = 0.0346m

  RMS Y = 0.0396m

  RMS Z = **0.3803m**

  Un anàlisi detallat permet detectar que el culpable és principalment el punt 304 amb un residu en cota de 1.66m, i.e. 35cm menys que l'error groller introduït. Part de l'error ha quedat escampat en altres punts de recolzament que sobrepassen els 10cm de residu, però en sentit contrari. El test de *data snooping* implementat actualment a *ACX* detecta sense problemes l'error groller.

  Pel que fa a l'error introduït al punt de canemàs, l'RMS no posa de manifest l'error groller per la gran quantitat d'observacions fotogramètriques a la xarxa. El residu fotogramètric de 17.46μm en x pel punt 7039008 a la imatge 8038 delata l'error groller, però de la mateixa manera que amb el recolzament, la resta de l'error introduït queda repartit per l'ajust.

- Ara li toca el torn a *Ceres*.

  Aplicant a *Ceres* el mètode general de mínims-quadrats i seguint el mateix procediment que amb ACX s'arriba a resultats equivalents: un residu de 17.46μm per la fotocoordenada x del punt de canemàs i un residu en cota pel punt de recolzament 304 de -1.66m.

- Aplicant mètodes robustos a *Ceres*.

  Afegim ara *Photo LossFunction* i *Control LossFunction* per provar de detectar els residus que "es perden" per la xarxa. Ambdues *loss functions* es basen en el mètode robust implementat a Ceres *Huber loss function* que és parametritzable a través d'un escalar *a*. Per aquesta simulació es fixa el valor del paràmetre *a* al valor que es recomana a [2]: *a = 1.0*.

  El residu del punt de recolzament és de 1.94m. Una diferència de 6cm és assumible dins de la distribució normal tenint en compte que la desviació estàndard de les observacions és de 4cm. Paral·lelament, es veu com el residu de la fotocoordenada x del punt 7039008 és de -21.63μm. De la mateixa manera que abans, una diferència respecte de l'error groller introduït de 1.63μm entra dins de la distribució normal ja que la desviació estàndard per observacions fotogramètriques és de 2μm.

5.4.5 Eficiència

L'eficiència és la capacitat de fer que l'ajust assoleixi convergència amb el mínim cost de computació. L'ús d'un mètode suficientment rigorós com per avaluar correctament l'eficiència s'escapa de l'abast d'aquest treball ja que implicaria un estudi en base a la teoria de complexitat computacional. Una opció seria fer un seguiment del nombre d'operacions d'assignació, comparació, accés i operacions aritmètiques. A la pràctica, una primera aproximació per tenir una

idea de l'eficiència és mesurant el temps d'execució suposant que les condicions de mesura es mantenen constants. És a dir, en absència de factors externs, la diferència en temps d'execució serà principalment deguda a la diferència d'eficiència entre softwares.

Per dur a terme les simulacions es fa servir un mateix ordinador amb plataforma de 64bits, CPU de doble nucli Intel Core i5 a 2.5GHz i 4GB de RAM.

Es prendrà com a referència el software vigent. L'eina de generació de dades sintètiques permetrà obtenir simultàniament dos ajustos idèntics en els formats propis de cada software.

5.4.5.1 Criteris de convergència

En aquest apartat sí serà necessari establir un criteri de convergència comú a ambdós softwares. De no ser-ho, un dels softwares faria iteracions innecessàries mentre l'altre podria aturar-se abans d'assolir convergència.

La dificultat de comparar l'eficiència d'*ACX* amb la del mòdul *Ceres* rau en el fet que no comparteixen la mateixa estratègia de resolució del sistema no lineal. *ACX* fa servir el mètode de Newton i a *Ceres* es pot escollir entre *Levenberg-Marquardt* i *Dog leg*. A cada estratègia es fa servir un *pas* d'una mida determinada i, per tant, el nombre d'iteracions no pot ser adoptat a cegues com a criteri de convergència.

El criteri de convergència comú és l'estabilització dels paràmetres. Ambdós softwares tenen possibilitat d'establir un valor de correccions als paràmetres per sota del qual s'atura el sistema iteratiu. Malgrat tot, el criteri és lleugerament diferent:

- *Convergence stop criterion d'ACX:* el criteri de convergència ve donat per un llindar escollit per l'usuari. El llindar és un nombre real anomenat *THR* (*threshold*). ACX aturarà el procés iteratiu en cas de que es compleixi alguna de les condicions següents:

  RMS de correccions als paràmetres < *THR i*

  Màxima correcció als paràmetres < *3 THR*

- *Ceres parameter tolerance:* si l'usuari escull un valor no nul pel valor de la tolerància *ParTol*, llavors l'ajust s'atura en cas de que:

$$\frac{\| \Delta x \|}{\| x \| + ParTol} < ParTol$$

  on ||x|| és el mòdul del vector de paràmetres i ||Δ*x*|| és el mòdul del pas per cada iteració del sistema no lineal [2].

Al moment de redactar aquesta memòria, l'ús del criteri de convergència de la versió de producció d'ACX no és operatiu. Així doncs ha estat necessari adoptar una alternativa menys elegant que passa per fer un anàlisi previ de les correccions per cada iteració:

1. S'ha executat el bloc amb cada software sense criteris de convergència permetent que faci un nombre d'iteracions àmpliament suficient com per assegurar la convergència
2. Un cop han acabat ambdues execucions, s'ha examinat el *log* que extreu cada un per analitzar les variacions del vector de correccions al llarg del procés iteratiu.
3. S'ha escollit un valor determinat de canvi en el vector de correcció als paràmetres i s'ha consultat el nombre d'iteracions necessàries per assolir-lo. El valor escollit és 10^{-8} per tal de garantir que convergirà per sota de la resolució de les coordenades objecte de sortida que és de 10^{-6}.

4. S'ha tornat a executar el bloc amb cada software, però ara imposant els nombres màxims d'iteracions obtinguts al punt anterior.

5.4.5.2 Simulacions

Per fer les simulacions s'ha dissenyat una configuració de vol prou gran com per que les diferències en temps d'execució puguin ser atribuïbles tant com sigui possible a la part central de càlcul (resolució de sistemes lineals, estratègia del sistema no lineal, mètodes numèrics aplicats a les operacions matricials, etc.). El límit de dimensions pel bloc vindrà donat pel fet de que ambdós programaris l'han de poder resoldre. A partir de diferents proves s'arriba a la conclusió de que ACX no és capaç d'admetre blocs tan grans com Ceres.

El bloc sintètic consta de 2000 imatges amb solapament longitudinal del 60% i transversal del 60%. Les orientacions externes es belluguen uns 2º de la posició nadiral utilitzant soroll gaussià. S'aplica un offset d'antena determinat. Les imatges simulades imiten la geometria d'una càmera digital de 120mm de distància focal amb una mida de píxel sobre el terreny resultant de 7.5cm. S'inclouen distorsions a les imatges que s'absorbiran automàticament per mitjà d'un grup d'autocalibratge d'en *Ebner*. L'ajust resultant consta de 136605 grups d'observacions, 24030 punts de canemàs i 35 punts de recolzament observats.

- Simulació T05a:
    - Resultats amb *ACX:*
        - Nombre d'iteracions: 4
        - Temps d'execució: 3min 10s
    - Resultats amb *Ceres* adoptant l'estratègia *Levenberg-Marquardt:*
        - Nombre d'iteracions: 11
        - Temps d'execució: 1min 7s
    - Resultats amb *Ceres* adoptant l'estratègia *Dog-Leg:*
        - Nombre d'iteracions: 9
        - Temps d'execució: 0min 57s

Amb l'eina *Compara* s'han avaluat les diferències de coordenades ajustades pels punts objecte. Les diferències obtingudes són inferiors a la resolució.

Queda pendent l'avaluació de l'eficiència d'ambdós softwares en dos extrems:

- Augmentant el nombre de paràmetres: augmentant el nombre d'imatges i de punts objecte i mantenint el nombre d'observacions. Es posa a prova la resolució del sistema normal ja que les dimensions de les matrius corresponents únicament depenen del nombre de paràmetres.
- Augmentant el nombre d'observacions: augmentant el nombre de feixos de cada punt objecte, és a dir, augmentant el nombre de fotocoordenades. Ho farem augmentant el solapament longitudinal i transversal i mantenint constant el nombre de punts i imatges.

5.4.6 Generalitat

Segons [32] un software genèric és aquell que permet considerar un ampli ventall de models matemàtics i de parametritzacions possibles. En aquest estadi de desenvolupament, el nou mòdul

Ceres considera únicament el cas fotogramètric de perspectiva nadiral segons la tècnica d'orientació integrada de sensors:

- Es consideren les següents observacions: fotocoordenades, punts de recolzament, mesures *GPS*, mesura d'un offset d'antena i pseudo-observacions d'un grup de paràmetres d'autocalibratge. Es tracta d'un subconjunt de les observacions que considera *ACX*.
- En quant a paràmetres considera punts objecte, orientacions externes, un offset d'antena i paràmetres d'autocalibratge. Igual que en el cas de les observacions també es tracta d'un subconjunt de paràmetres respecte dels que considera *ACX*.
- Pel que fa a models, *ACX* contempla molts més casos que el nou mòdul *Ceres*. Per exemple: espais de paràmetres no euclidians (projecció UTM), geometria de sensors multiespectrals, xarxes geodèsiques i un llarg etcètera.
- L'avantatge principal en generalitat respecte d'*ACX* ve donada per l'ampli ventall de parametritzacions possibles. *Ceres* compta a data d'avui amb tres estratègies de resolució del sistema no lineal a escollir (*Levenberg-Marquardt*, *Traditional dogleg* i *subspace dogleg*), diversos mètodes de resolució del sistema lineal (*Cholesky*, *Schur*, etc.) . En canvi, *ACX* està molt més limitat en aquest sentit.

Actualment, el nou mòdul *Ceres* és significativament menys genèric que *ACX* en la majoria d'aspectes. Abans d'incorporar nous models per augmentar la generalitat del software caldrà optimitzar la flexibilitat.

5.4.7 Flexibilitat

Segons [32] un software flexible és aquell que permet incorporar nous models matemàtics. En aquest sentit s'han determinat tres línies de millora en la flexibilitat del nou mòdul.

1. **Diferenciació automàtica:** És el punt més fort en quant a flexibilitat. Permet implementar nous models amb un esforç mínim. Només cal definir el model matemàtic i *Ceres* s'encarrega de linealitzar el sistema.
2. **Multi-parametrització "*on the fly*":** Important millora de flexibilitat respecte d'*ACX*. *Ceres* ja té implementats molts mètodes de resolució i moltes opcions per escollir la combinació ideal per cada cas. Per provar un nou model, nova estratègia, etc. només cal escollir-la d'entre les moltes disponibles. Les que hi ha disponibles són tecnologia d'avantguarda ja que *Google* s'hi dedica contínuament i rep feedback d'usuaris *Ceres* d'arreu del món. En cas de voler utilitzar un mètode concret no disponible a *Ceres*, es podria enllaçar amb la llibreria corresponent. En aquest sentit, la universalitat d'ús del llenguatge en què està programat *Ceres* (C++) és un clar avantatge.
3. **Implementació àgil de nous models:** La flexibilitat vindrà donada en gran mesura per la facilitat per incorporar nous models. Ara mateix el nou mòdul és poc flexible en aquest sentit perquè per afegir un nou model cal modificar varis arxius de codi i capçaleres. La manera de millorar la flexibilitat consistirà en reorientar les classes per tal d'independitzar la definició de grups de paràmetres i observacions de la definició dels models. És a dir, crear classes pare *grup de paràmetres* i *grup d'observacions* i, paral·lelament, crear subclasses de les classes pare per cada un dels nous grups de paràmetres i observacions particulars relatius al nou model. Queda pendent amb la nova estructura de classes (TBD). És factible. Només cal disposar d'un programador amb bons coneixements de programació orientada a objectes.

Un cop resolta la limitació exposada al punt nº3, la incorporació de nous paràmetres, observacions i els models corresponents serà immediata.

5.5 Jugant amb *Ceres*

Per últim es mostren dues proves més que s'han considerat rellevants per avaluar les possibilitats que ofereix *Ceres*. Cal tenir present que per les simulacions no s'utilitza l'última versió del programari sinó 1.8.0[13].

La primera simulació servirà per ensenyar els efectes d'escollir una o altra estratègia de resolució del sistema no lineal i la segona permetrà comprovar quins són els límits de Ceres pel que fa a les dimensions màximes del problema.

5.5.1 Simulació S01. Quin mètode convergeix més ràpid?

Per la simulació S01 s'ha generat amb *AeroSint* un bloc sintètic de 120 imatges. No s'ha afegit soroll gaussià a les observacions, per tant es coneixen amb total exactitud els veritables valors dels paràmetres. Les aproximacions inicials s'han truncat a 1 m en l'espai objecte.

S'ha ajustat el bloc utilitzant tres estratègies de resolució del sistema no lineal diferents: *Gauss-Newton*, *Levenberg-Marquardt* i *Dog-leg*. Deixant de banda el temps d'execució, què no es pot comparar degut a l'ús de diferents softwares[14], es pot observar com el nombre d'iteracions és bastant diferent segons el mètode escollit.

A la figura 5.2 es mostra la convergència de la solució per les tres estratègies al llarg del procés iteratiu corresponent.

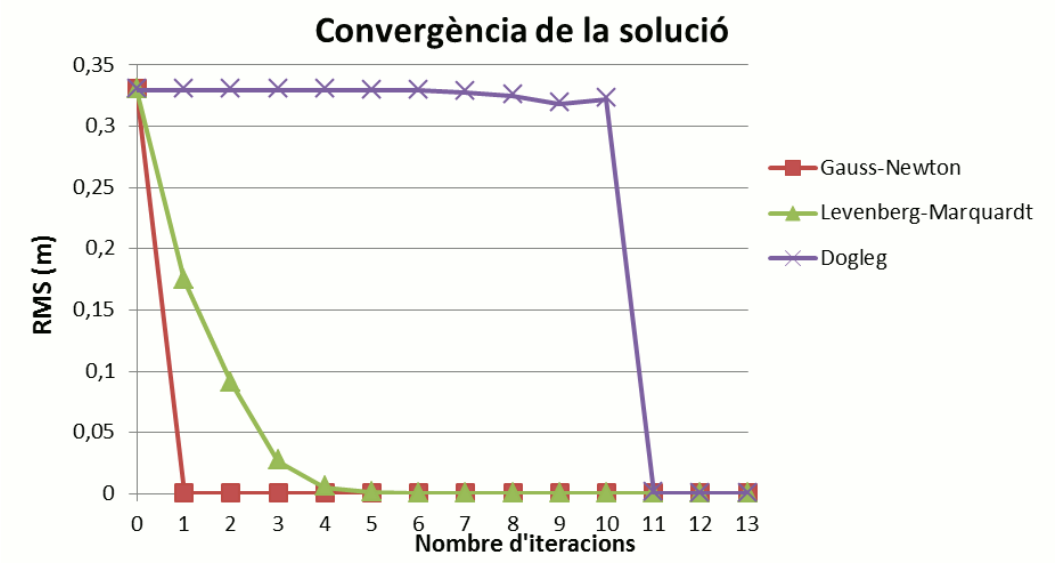


Figura 5.2

Es pot observar que el mètode de *Gauss-Newton* té el millor comportament per aquest bloc. El motiu de que *Dog-leg* passi a convergir "de sobte" és que a la iteració nº11 l'estratègia passa a adoptar un pas de *Newton*.

[13] A data del lliurament d'aquest projecte es troba disponible una versió més actualitzada de Ceres Solver, la 1.9.0.

[14] Pel mètode de *GN* s'ha utilitzat *ACX* i pels de *LM* i *DL* s'ha utilitzat *Ceres.*

A la figura 5.3 es mostra l'evolució del *pas* i el radi de la regió de confiança per l'estratègia *Dogleg* en el mateix problema anterior.

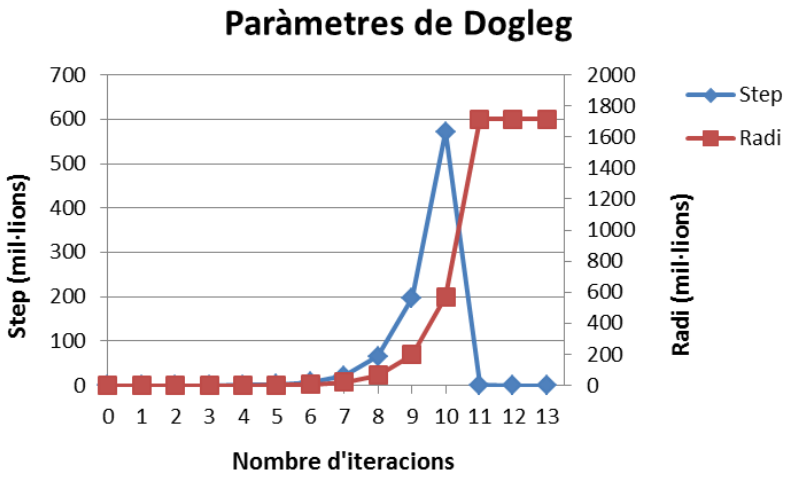


Figura 5.3

5.5.2 Simulació S02. Portant *Ceres* al límit

A data d'avui, la versió d'*ACX* de producció no admet més d'uns 200000 grups d'observacions que és equivalent a blocs d'entre 2000 i 4000 imatges depenent del solapament, densitat de punts per imatge, etc. Segons les proves efectuades en aquest treball, per blocs amb 60% de solapament longitudinal i 60% de solapament transversal el límit d'*ACX* és de l'ordre d'unes 2400 imatges. Baixant el solapament transversal al 30%, *ACX* passa a ser capaç d'ajustar més de 3000 imatges.

A continuació es dissenya el bloc més potent que s'ha aconseguit ajustar amb Ceres. Es tracta d'un bloc sintètic 9000 imatges amb solapament longitudinal del 60% i transversal del 60%. Les orientacions externes es belluguen uns 2º de la posició nadiral utilitzant soroll gaussià. S'aplica un offset d'antena determinat. Les imatges simulades imiten la geometria d'una càmera digital de 120mm de distància focal amb una mida de píxel sobre el terreny resultant de 7.5cm. S'inclouen distorsions a les imatges que s'absorbiran automàticament per mitjà d'un grup d'autocalibratge d'en *Ebner*. L'ajust resultant consta de 624420 grups d'observacions, 108091punts de canemàs i 96 punts de recolzament observats.

S'han executat les simulacions amb el mateix ordinador i els mateixos criteris de convergència establers a l'apartat *5.4.5 Eficiència*. La tolerància de paràmetres establerta és de 10^{-8}.

- Resultats amb *Ceres* adoptant l'estratègia *Levenberg-Marquardt:*
  - Nombre d'iteracions: 12
  - Temps d'execució: 13min 2s
- Resultats amb *Ceres* adoptant l'estratègia *Dog-Leg:*
  - Nombre d'iteracions: 10
  - Temps d'execució: 10min 50s

Amb la versió de Ceres Solver utilitzada en aquest treball, si es proven d'ajustar blocs fotogramètrics poc més grans que l'anterior, el programa retorna un error numèric degut a un desbordament de memòria a *cholmod*, una de les rutines de *SuiteSparse* que utilitza *Ceres*.

6 Conclusions

S'ha desenvolupat un nou mòdul basat en *Ceres* que permet resoldre eficientment blocs d'aerotriangulació utilitzant el mètode d'ajust de feixos en bloc i considerant observacions fotogramètriques, de recolzament, mesures GPS i offset d'antena. Addicionalment permet contemplar paràmetres d'autocalibratge d'Ebner que absorbirien possibles distorsions en les imatges. Paral·lelament, el generador de dades sintètiques *AeroSint* ha permès posar a prova el nou mòdul *Ceres* i avaluar els punts forts i les debilitats prenent com a referència el programari vigent, l'*ACX*.

6.1 Punts forts

Els principals punts forts del nou mòdul *Ceres* són l'eficiència, l'ús de memòria dinàmica, l'avantatge que suposa la diferenciació automàtica a l'hora de crear nous models, el tractament especialitzat de matrius disperses, i el suport continuat per part dels desenvolupadors de *Google*. A més, la possibilitat d'escollir multitud de combinacions de mètodes de resolució tant del sistema lineal com del no lineal, criteris de convergència, i un llarg etcètera permet trobar la parametrització que més bé s'adapti al problema concret. En quant a la robustesa, les poques proves que s'han fet amb mètodes robustos són molt esperançadores. Amb poc més de dues línies de codi es poden implementar uns mètodes robustos "que fan el que han de fer" gràcies a la simplicitat de les llibreries *Ceres*.

Alguns dels blocs simulats mostren una petita millora en l'estabilitat. Així i tot la millora que proporcionen els mètodes de *Dog leg* i *Levenberg-Marquardt* comparada amb la del mètode de *Gauss-Newton* no és tan gran com s'esperava.

En quant a l'exactitud, per totes les simulacions efectuades en aquest treball (aquelles que hagin assolit convergència) s'ha obtingut una discrepància nul·la o quasi nul·la respecte d'*ACX*. La conclusió és que s'ha aconseguit emular amb fidelitat el comportament d'*ACX*.

A falta d'implementar un anàlisi estocàstic rigorós que permeti obtenir informació sobre les desviacions a posteriori dels paràmetres ajustats, els residus de les observacions mostren un comportament idèntic al d'*ACX*.

En comparació amb *ACX*, el nou mòdul *Ceres* es presenta com una alternativa eficient i que per tant podria implantar-se a la cadena de producció de l'*ICGC* per tal d'agilitzar-la. Un dels motius principals d'aquesta millora és el tractament optimitzat de matrius disperses gràcies a que *Ceres* utilitza llibreries especialitzades com *SuiteSparse*.

6.2 Propostes de millora

A data del lliurament d'aquest projecte la flexibilitat del codi es veu afectada de manera considerable per culpa de l'estructura de classes implementada. Caldria re-implementar el codi en base a l'estructura jeràrquica proposada i així aprofitar tot el potencial de la programació orientada a objectes per obtenir un mòdul més flexible.

L'anàlisi estocàstic està inacabat. Fer un anàlisi acurat de la informació estocàstica passa per consultar la matriu de variància-covariància de la solució. Per tal d'implementar-lo es recomana aprofitar la classe covariance.h [38] que *Google* ha incorporat recentment a la nova versió de Ceres.

Amb les proves efectuades en aquest treball no s'ha arribat a la mateixa conclusió que arriben a [18] en relació al millor mètode per resoldre l'ajust de feixos. Allà es defensa que amb l'estratègia *Dog leg* s'obtenen resultats àmpliament millors que amb *Levenberg-Marquardt*. En aquest treball algunes simulacions demostren que no sempre és així. De fet, aquí la conclusió a la que s'arriba és

que amb *Dog leg* s'aconsegueix una petita millora en eficiència per blocs molt grans a expenses de més inestabilitat en blocs petits o amb observacions que tinguin desviacions properes a zero.

En ajustos que parteixen d'aproximacions inicials bones, *ACX* presenta una convergència significativament més ràpida que Ceres perquè utilitza l'estratègia de *Gauss-Newton* per resoldre el sistema no lineal. Considerem que això és una limitació de *Ceres* ja que només permet triar entre els mètodes de *Levenberg-Marquardt* i *Dog leg*.

Malgrat a [2] es recomana que pel problema concret de l'ajust de feixos s'utilitzi el mètode de *Schur* per obtenir la solució al sistema linealitzat, en aquest treball s'utilitza el mètode de *Cholesky* degut a que no ha estat possible depurar un error de compilació relacionat amb una de les llibreries que utilitza *Ceres*. Si es volgués millorar l'eficiència del mòdul es recomana depurar la compilació de *Ceres* per tal de detectar el problema.

Per últim, es recomana que abans d'afegir més models, en especial els referents a observacions inercials, es re-implementin les matrius de rotacions de les equacions de col·linealitat utilitzant quaternions en comptes d'angles d'Euler. D'aquesta manera es pretén aconseguir una major estabilitat del model funcional.

7 Bibliografia

[1] Colomina I. *Structural aspects of hybrid networks in geodesy and photogrammetry*. Apèndix A: *The photogrammetric network case*. Tesi doctoral, Departament de Matemàtica Aplicada i Anàlisi, Universitat de Barcelona, 1993.

[2] Agarwal S., Mierle K. *Ceres Solver: Tutorial & Reference*. Google Inc, 2012.

[3] Lourakis M., Argyros A. *The design and implementation of a generic sparse bundle adjustment software package based on the Levenberg-Marquardt algorithm*, 2004

[4] Konolige K. *Sparse sparse bundle adjustment*, 2010

[5] Triggs B., McLauchlan P., Hartley R., i Fitzgibbon A. *Bundle Adjustment - A Modern Synthesis*. In Proceedings of the International Workshop on Vision Algorithms: Theory and Practice, p. 298–372, 1999.

[6] Baron A., Kornus W., Talaya J. *ICC experiences on Inertial / GPS sensor orientation*. Institut Cartogràfic de Catalunya, 2003.

[7] Blázquez M., Colomina I. *Fast AT: A simple procedure for quasi direct orientation*. Institut de Geomàtica. Abril, 2012.

[8] Blázquez M. *A systematic approach to airborne sensor orientation and calibration: methods and models*, maig de 2012.

[9] Colomina I., Blázquez M., Navarro J.A., Sastre J. *The need and keys for a new generation network adjustment software*, 2012.

[10] Klingner B., Martin D. i Roseborough J. *Street View Motion-from-Structure-from-Motion*. International Conference on Computer Vision (ICCV), 2013

[11] Agarwal S., Snavely N., Simon I., Seitz S., Szeliski R. *Building Rome in a Day*. A Proceedings of the International Conference on Computer Vision (ICCV), 2009.

[12] Zhao L., et al. *ParallaxBA: Bundle Adjustment using Parallax Angle Feature Parametrization*. Article presentat a la *International Journal of Robotics Research (IJRR)*. Agost, 2013.

[13] Lee J. C. *Project Tango*. ATAP (Advanced Technology and Projects). Google Inc., 2013. https://www.google.com/atap/projecttango/#project

[14] Rodríguez, J.J. *Ajuste de observaciones.* Edicions UPC, Barcelona, 2002.

[15] Madsen K., Nielsen H.B., Tingleff O. *Methods for nonlinear least squares problems*. Informatics and Mathematical Modelling Technical University of Denmark. Segona edició, abril de 2004.

[16] Eade E. *Gauss-Newton / Levenberg-Marquardt Optimization*, març de2013.

[17] Press, W.H. et al., *Numerical Recipes in C. The art of scientific computing.* Cambridge University Press, Cambridge, 1989.

[18] Lourakis M., Argyros A. *Is Levenberg-Marquardt the most efficient optimization algorithm for implementing bundle adjustment?*. International Conference on Computer Vision (ICCV), 2005.

[19] Gallier J. *The Schur Complement and symmetric positive semidefinite (and definite) matrices*, desembre de 2010.

[20] Mostafa H. *Symbolic differentiation*. Code Project, març de 2008. http://www.codeproject.com/Articles/13731/Symbolic-Differentiation

[21] Piponi D. *Automatic differentiation, C++ Templates and Photogrammetry*, setembre, 2004.

[22] Vasantha W. B., Smarandache F. *Dual numbers*. Capítol 2. Ohio, 2012

[23] Lourakis M., Argyros A. *SBA: A Software Package for Generic Sparse Bundle Adjustment*, març de 2009.

[24] Lourakis M. *Overview presentation of SBA and BA*

[25] Institut für Photogrammetrie Stuttgart http://www.ifp.uni-stuttgart.de/publications/software/openbundle/index.en.html

[26] Stallmann D. *DGAP Notes*. Març, 2008.

[27] *Institute National de l'Information Géographique et Forestière http://logiciels.ign.fr/?Micmac*

[28] Pierrot-Deseilligny M., Paparoditis N. *A multiresolution and optimization-based image matching approach: An application to surface reconstruction from SPOT5-HRS stereo imagery. ISPRS Workshop on Topographic mapping from space (with special emphasis on small satellites).* Ancara, febrer, 2006.

[29] Hartley R. I. *An object-oriented approach to scene reconstruction*, 1996

[30] Snavely N., Seitz S.M., Szeliski R. *Modeling the World from Internet Photo Collections*. International Journal of Computer Vision, octubre de 2007.

[31] Snavely N., Seitz S., Szeliski R. *Photo tourism: exploring image collections in 3D*. ACM Transactions on Graphics (Proceedings of SIGGRAPH 2006), 2006.

[32] Colomina I., Navarro J., Térmens A. *GeoTeX: a general point determination system*. *Institut Cartogràfic de Catalunya*, 1992.

[33] Buill, F., Núñez, M.A., Rodríguez, J.J. *Fotogrametría analítica*. Edicions UPC, Barcelona, 2003.

[34] *Python 2.7.7 documentation*. https://docs.python.org/2/index.html

[35] Evans C. *YAML*. http://www.yaml.org/

[36] Bosch M. *Apunts d'informàtica del curs 2007-2008*. Departament de Matemàtica Aplicada i Anàlisi, Facultat de matemàtiques, Universitat de Barcelona, maig de 2008.

[37] Agarwal S., Mierle K. et al. *Ceres Solver*. Google Inc., 2014. http://ceres-solver.org/

[38] Ceres Solver source code: Mòdul de covariances de *Ceres Solver*:

https://ceres-solver.googlesource.com/ceres-solver/+/master/include/ceres/covariance.h

[39] Ceres Solver Google Groups: Weighting / Covariances. Google Groups:

https://groups.google.com/d/topic/ceres-solver/OobKeuuvXHE/discussion

[40] Ebner H. *Self calibrating block adjustment*. XIII Congress of the ISPRS, Com. III, Helsinki, 1976.

[41] Ceres Solver Google Groups: Weighted non-linear least squares. Google Groups:

https://groups.google.com/d/topic/ceres-solver/1yAHTEto9gA/discussion

[42] Ceres Solver Google Groups: Photogrammetric approach

https://groups.google.com/d/topic/ceres-solver/yN9syhsNFRs/discussion

8 Agraïments

A l'Albert Prades i Valls, el meu tutor de projecte, per confiar en mi un altre cop i per cedir-me el codi del seu generador de soroll gaussià que m'ha vingut de perles pel mòdul *AeroSint*.

Al Joan Arnaldich i Bernal, el tutor de part de l'*ICGC*, per classes particulars de programació i, especialment, per les incomptables hores que ha tret del seu temps per dissenyar els prototipus i ensenyar-me a utilitzar-los. Sense ell la *Ceres* ni tan sols compilaria.

A la Dra. Assumpció Térmens, per les mil i una explicacions sobre els models matemàtics de l'*ACX*.

A l'Anna Gabarrón Comadran per recolzar-me amb el projecte i ajudar-me amb la memòria (escrivint fórmules amb l'editor d'equacions, correccions vàries, etc.). Sense ella no m'hagués decidit a lliurar encara.

A l'*ICGC* per l'ús de les infraestructures (documentació, eines, programari, estació de treball, etc.) i dades de blocs fotogramètrics de producció que m'han permès començar les proves preliminars abans de desenvolupar *AeroSint*.